Diogo Alexandre Nascimento Alves

Licenciatura em Ciências da
Engenharia Electrotécnica e de Computadores

Desenvolvimento de um Sistema de Apoio ao Arquivo e à Gestão de Blocos Histológicos

Dissertação para obtenção do Grau de Mestre em
Engenharia Electrotécnica e de Computadores

Orientador: José Manuel Matos Ribeiro da Fonseca, Professor Auxiliar com Agregação, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Co-orientador: André Teixeira Bento Damas Mora, Professor Auxiliar, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2018

Desenvolvimento de um Sistema de Apoio ao Arquivo e á Gestão de Blocos Histológicos

Agradecimentos

Em primeiro lugar, gostaria de agradecer aos professores José Manuel Fonseca e André Damas Mora, não só pela oportunidade concedida de trabalhar e desenvolver este projeto sobre a sua tutela e orientação, como por todo o apoio disponibilizado, e a paciência demonstrada, neste mesmo, ou nas unidades curriculares que frequentei ao encargo destes docentes. Foram, sempre que possível, uma tremenda ajuda, e um ouvido amigo.

Gostaria de agradecer à instituição patológica para a qual este projeto se destina pela sua disponibilização de uma gaveta e 20 blocos histológicos, cedidos com o intuito de melhor poder simular as suas condições de trabalho, de modo a melhor poder concluir este projeto.

Gostaria de agradecer, em jeito de três tempos, a todos os meus amigos que me acompanharam ao longo dos anos na minha educação e crescimento enquanto estudante: a todos os meus amigos da universidade, que me acompanharam durante estes árduos anos de trabalho – gostaria de destacar, pela nossa maior afinidade, o David, a Inês, o Miguel, o Frederico, o José, o André e o Davidson - um muito obrigado pelo vosso apoio e amizade durante o curso, com um pequeno destaque para o David, que foi o meu companheiro de trabalho durante todo o mestrado, dissertação inclusive; ao Ricardo, ao Bruno, ao António, à Catarina e à Filipa, que me acompanham desde o ensino secundário – um muito obrigado pelo vosso apoio e amizade, e que a nossa família seja para sempre; ao Tiago, que me acompanha já desde o ensino básico, e que me auxiliou também no desenvolvimento deste projeto – um muito obrigado pelo teu apoio e amizade, e que, como tu dizes, nunca nos deixemos de falar. Espero que todas as vossas amizades para comigo sejam intermináveis.

Finalmente, gostaria de agradecer aos meus pais: mesmo que tudo corresse ao contrário do esperado ou do favorável, pelo meio de todos os obstáculos, sempre tentaram remar contra a maré para nos manter à tona, e sempre deram o vosso máximo para que eu pudesse ter as mesmas oportunidades que todos os outros. Foi através da vossa educação que, para melhor ou pior, para bem e para o mal, me tornei no Homem que sou hoje. Sem o vosso esforço conjunto, Diogo Alexandre Nascimento Alves como o conheço, não existiria, e por isso, um muitíssimo obrigado. Estarão sempre no meu coração.

Resumo

No mundo da Medicina, uma biopsia é um exame comum que consiste na remoção de um pedaço pequeno de tecido, com o objetivo de averiguar a existência ou não de doença no paciente. Por questões de saúde e regulamento, as amostras retiradas têm de ser devidamente preservadas e armazenadas em pequenos blocos histológicos, que possuem na sua composição as informações relevantes para identificar o paciente.

Tal esforço é difícil de realizar devido ao volume intenso de exames por amostra e o facto de ser um processo manual. Se se perder um bloco, pode despoletar uma situação muito inconveniente tanto para instituto como paciente, o que ninguém pretenderá.

De modo a se tentar resolver este dilema, a solução passou pela criação de um sistema automatizado para auxiliar o patologista no registo e gestão dos blocos: o sistema faz o registo automático de várias gavetas de blocos simultaneamente em formato eletrónico com o auxílio de um *scanner*, cria e preenche uma base de dados informática com as informações retiradas, e serve também como centro de requisição e gestão da localização dos blocos.

Com o desenvolvimento deste sistema, pretende-se diminuir o erro humano associado ao processo de registo e o tempo perdido, bem como aumentar a eficácia e segurança do mesmo.

Palavras-chave: Bloco histológico, reconhecimento ótico de caracteres, digitalização, processamento de imagem, *Windows Image Acquisition*.

Abstract

In the Healthcare world, a biopsy is a routine exam consisting in the removal of a small piece of tissue, for verifying whether the patient is diseased. Due to health and regulation concerns, the samples taken must be properly preserved and stored in small tissue cassettes, which contain relevant information to identify the patient.

Such an effort is hard to accomplish due to the sheer volume of tests per sample, and the fact that it is a manual process. If one cassette is lost, it could set off a very inconvenient situation for both institute and patient, which nobody will want to be a part of.

In order to solve this dilemma, the solution chosen was to create an automatized system, to help the pathologist in the recording and management of the cassettes: the system is able to perform the automatic record of several drawers of cassettes simultaneously in electronic format with help from a scanner, to create and fill an informatic database with the information retrieved, and also to double as a request and management center for the cassettes and their localizations.

With the development of this system, the goal is to decrease the lost time and the human error associated with the recording process, as well as increasing its' efficiency and safety.

Keywords: Tissue cassette, optical character recognition, scanning, image processing, Windows Image Acquisition.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

IVT - Instituto Virtual da Tese

API - Application Programming Interface

WIA - Windows Image Acquisition

OS - Operating System

DLL - Dynamic-link Library

ISIS - Image and Scanner Interface Specification

SANE - Scanner Access Now Easy

OCR - Optical Character Recognition

SDK - Software Development Kit

PDF - Portable Document Format

GUI - Graphical User Interface

HTML - Hypertext Markup Language

1

Introdução

Este capítulo é dividido em quatro partes distintas, para efeitos de introdução ao projeto desenvolvido: numa primeira parte, o projeto de tese a realizar será contextualizado e apresentado: em que âmbito da tecnologia/sociedade/investigação/etc. se insere, que razões movem esse sector ou que atividades são feitas nesse contexto.

Na segunda parte, será apresentada a motivação que levou à idealização deste projeto, bem como o objetivo final que ele pretende solucionar ou facilitar.

Na terceira parte, será apresentada a proposta de trabalho que visa realizar esse mesmo objetivo.

Finalmente, na quarta parte será apresentada a estrutura da tese desenvolvida, capítulo a capítulo, de uma forma sintetizada e generalizada, de modo a que qualquer leitor consiga perceber, livre de confusões, o conteúdo que cada capítulo aborda.

1.1 Contextualização Temática

No mundo da Medicina, uma biópsia é uma técnica frequentemente utilizada no âmbito da oncologia e da patologia para determinar a existência ou não de um cancro num paciente: dito de um modo simplificado, trata-se da extração de uma amostra de tecido de um órgão de um ser vivo, geralmente efetuada por um cirurgião, para posterior examinação sob microscópio, esta efetuada por um patologista. [1]

Antes de ser examinada, a amostra tem de passar por um processamento, constituído da seguinte forma [2]:

- A primeira etapa no processo de pré-análise é a de fixação, isto é, de preservação química, no qual a amostra é embebida em formol a 10% o mais depressa possível após a sua extração, de modo a evitar lesões no tecido que possam impossibilitar a sua análise;
- De seguida a amostra é incluída em parafina líquida, a qual é arrefecida até um estado sólido, reminiscente a uma cera ou resina, que passa a englobar a amostra no seu interior. Este processo de encapsulamento em parafina é realizado com a amostra colocada dentro de um bloco de plástico que a irá conter, denominado por cassete para biópsia, ou bloco de parafina. Este pode ser visualizado na Figura 1.1 abaixo:

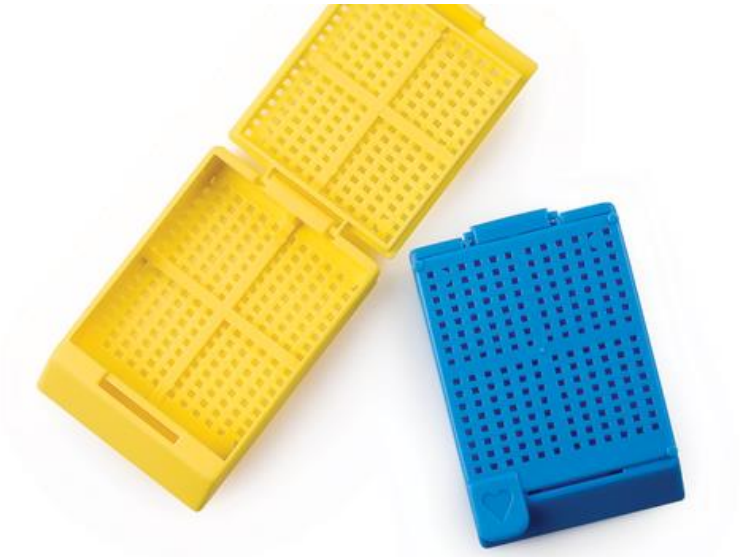


Figura 1.1 - Exemplo de blocos de parafina, como apresentados no *site* da Leica Biosystems [3]

Finalizado o processamento da amostra e descartada a parafina em excesso, que não contenha partes da amostra, o resultado final será um bloco retangular de parafina, com a amostra no seu interior. Neste momento, a amostra encontra-se finalmente preparada para ser cortada por um micrótomo (visualizado abaixo, na Figura 1.2): um aparelho que faz cortes microscópicos, fatiando a amostra em várias secções de espessura micrométricas que serão então montadas entre duas lâminas de vidro, possibilitando a sua análise a microscópio.

Figura 1.2 – Micrótomo deslizante modelo SM2010 R, da marca Leica [4]

Tanto as lâminas como os blocos, quando não se encontram em uso, são armazenados em gavetas específicas, cada um com a sua identificação e codificação específica. Tendo em conta que um único paciente pode chegar às centenas de lâminas por amostra, e um único instituto pode ter vários pacientes e efetuar vários testes por dia, durante vários dias, torna-se necessário armazenar todo este material; a tarefa torna-se ainda mais complicada quando se tem de identificar de um modo simples e eficaz o suficiente para os trabalhadores do instituto poderem, entre si, retirar, analisar e devolver ao seu lugar as amostras corretas a serem analisadas, devolvendo-as depois ao seu lugar, onde devem permanecer durante pelo menos 10 anos. [5]

1.2 Motivação e Objetivo

É com o problema acima descrevido em mente que um instituto patológico – cujo nome verdadeiro, por questões de ética, não será revelado, e será ao longo desta tese referido como o Instituto Virtual da Tese, ou IVT como abreviatura – procurou auxílio junto do professor José Manuel Fonseca, de modo a conseguir solucioná-lo.

Dentro do IVT, o registo dos blocos histológicos é realizado inteiramente sem qualquer auxílio informático: os blocos são guardados em gavetas estreitas e compridas, que, por sua vez, são inseridas em armários, que armazenam várias destas gavetas. É necessário acompanhar com atenção a entrada e saída de blocos, para que nenhum se perca, não só pelo pesadelo logístico, como também pelas possíveis repercussões a nível da saúde. Tendo em conta o volume de testes que um patologista pode ter de fazer, e a quantidade de blocos que existem, a serem movidos de um lado para o outro, torna-se muito fácil perder a noção dos seus paradeiros.

Percebendo a natureza precária e delicada à volta do processamento dos blocos, torna-se então necessário encontrar uma solução que permita aos patologistas um melhor e mais simples método de trabalho.

1.3 Proposta

De modo a ajudar a uma melhor organização, maior eficiência de trabalho e minimização de erros de logística no armazenamento, requisição e examinação das amostras contidas nos blocos de parafina do IVT, é proposto, como objetivo de tese: a criação de um dispositivo na ordem de um *scanner* que contenha na sua estrutura uma ranhura ou abertura que permita a inserção de várias gavetas de armazenamento de blocos de parafina, com o fim de digitalizar a identificação nelas contidas para a sua introdução numa espécie de base de dados, a qual será controlada por um *software*, que se irá encarregar da ligação *scanner*-computador, tratamento da imagem digitalizada e gestão das identificações dos blocos de parafina.

Para a construção do dispositivo digitalizador dos blocos, será necessário adquirir um *scanner*, com tampa amovível, e adaptá-lo, criando um compartimento com dimensões suficientes para alojar as gavetas e digitalizar os blocos com a melhor fidelidade de imagem possível.

O *software* a desenvolver será o responsável pelo processamento da imagem captada e por decifrar os caracteres digitalizados, registando-os o de seguida numa base de dados, a qual será também gerida pelo *software*, que irá permitir ao utilizador corrigir qualquer possível erro de registo ao adicionar manualmente uma entrada na base de dados e servir também como centro de requisição e devolução de blocos de parafina, para que qualquer funcionário consiga informar/ser informado pelos colegas sobre onde se encontra um qualquer bloco.

1.4 Estrutura de Tese

Esta dissertação está organizada em 6 capítulos distintos: Introdução, Estado da Arte, Fundamentos Teóricos, Metodologia e Implementação, Resultados e Discussão e, por último, Conclusões e Trabalho Futuro.

O primeiro capítulo, Introdução, é o capítulo atual, onde se introduz o contexto onde se insere o problema a abordar ao leitor, e onde se expõe a motivação e objetivo da realização da tese, bem como a proposta pensada para solucionar o problema existente.

O segundo capítulo, Estado da Arte, irá apresentar ao leitor um pequeno estudo de mercado em relação às tecnologias a utilizar no desenvolvimento da tese, nomeadamente na área de *softwares* de reconhecimento ótico de caracteres, de *scanners* e de eventuais produtos pré-existentes semelhantes à proposta de tese.

O terceiro capítulo, Fundamentos Teóricos, serve para explicar ao leitor os conceitos base por detrás do desenvolvimento da tese, de modo a que este perceba melhor, sem ter de ser instruído, o que foi feito, como foi feito, e porque foi feito de uma maneira e não de outra. De um modo geral, serão explicados melhor os conceitos de processamento de imagem, reconhecimento ótico de caracteres, e o algoritmo utilizado para resolver o problema, bem como os conceitos no qual ele se baseia.

No quarto capítulo, Metodologia e Implementação, é exposto e esclarecido o processo de construção e desenvolvimento do projeto proposto: construção do protótipo da caixa envolvente, onde serão inseridas as gavetas para digitalização; descrição do *software* principal, desenvolvido pelo autor, e das suas componentes constituintes e quais os *softwares* adicionais utilizados para facilitar a construção e bom funcionamento do *software* principal.

No quinto capítulo, Resultados e Discussão, são apresentados os resultados obtidos dos testes realizados após a conclusão da implementação do projeto e, posteriormente, é discutida a validação da proposta ou não, mediante os resultados obtidos.

Finalmente, no sexto capítulo, Conclusões e Trabalho Futuro, são apresentadas algumas considerações mais aprofundadas sobre os resultados obtidos, nomeadamente em relação à conclusão do objetivo, à validação da proposta, aos aspetos positivos e negativos do desenvolvimento, às dificuldades encontradas, e a possíveis trabalhos futuros.

2

Estado da Arte

Neste capítulo, será exposta a pesquisa realizada pelo autor em relação ao Estado da Arte: esta foi feita com o intuito de perceber, dentro do contexto da proposta de tese, o estado atual das tecnologias envolventes e de produtos existentes, e como é que estes podem influenciar o desenvolvimento deste projeto.

Dum modo mais específico, procedeu-se a uma investigação sobre 4 pontos: o contexto atual dos *scanners*, os produtos existentes e as suas funcionalidades; o contexto atual dos protocolos de aquisição de imagem, os produtos existentes e uma comparação de funcionalidades entre si; o contexto atual do reconhecimento ótico de caracteres, os produtos existentes que empreguem essa mecânica e o seu possível uso no desenvolvimento deste projeto e, por último, a averiguação de produtos semelhantes à construção final idealizada na proposta de tese, e como esses produtos poderão inviabilizar ou não o desenvolvimento desta tese.

2.1 Tecnologias de Apoio

2.1.1 Scanners

Um *scanner* é um dispositivo que captura imagens, texto escrito e/ou impresso e objetos e digitaliza-os, criando uma representação digital do que se encontra na sua área de digitalização – uma superfície de vidro, a qual é iluminada, a maior parte das vezes, contra um fundo branco no lado oposto, por uma luz que percorre a totalidade da secção de vidro repetidas vezes.

Existem vários tipos de *scanners*, sendo que a variação mais comum é o *flatbed scanner* ou *scanner* de mesa (exemplo visualizado abaixo na Figura 2.1), muitas vezes utilizado em escritórios ou em domicílios. Sendo este o tipo de *scanner* comercialmente mais disponível, e mediante a impossibilidade plausível de se construir um de raiz, um objetivo passa pela aquisição de um *scanner* deste tipo.

Figura 2.1 - *Scanner* da Epson do modelo Perfection V19 [6]

Muitos *scanners* atualmente já são expedidos com *softwares* de reconhecimento ótico de caracteres embutidos; no entanto, muitos são programas executáveis em estilo de "caixa preta", isto é, o consumidor não tem, em condições normais e legais, maneira de olhar para como o programa é escrito e desenvolvido, não podendo interligar o *software* embutido com outro qualquer da sua escolha e/ou autoria.

Como a proposta de tese passa pelo desenvolvimento de um *software* que sirva de "centro de operações" de todo a gestão e aquisição digital das informações dos blocos, a existência ou não de um *software* pré-existente não será entrave para o desenvolvimento da tese.

2.1.2 Protocolos de Aquisição de Imagem

Como explicado acima, um *scanner* irá criar uma imagem digital do que quer que esteja inserido na sua área de digitalização e, posteriormente, envia esta imagem ao utilizador, através da conexão *scanner*-computador, para este a visualizar. Esta conexão que permite o envio das imagens digitalizadas é conseguida através do uso de um protocolo de aquisição de imagem.

Um protocolo deste género pode ser constituído por uma API, que através do conjunto das suas sub-rotinas e funções, permite a comunicação entre dois ou mais componentes diferentes de um sistema, neste caso, entre *scanner* e computador.

Nos subcapítulos abaixo, serão explorados alguns destes protocolos e, no final, comparados entre si, de modo a perceber qual será a melhor escolha para implementar neste projeto.

2.1.2.1 *Windows Image Acquisition* – WIA

Desenvolvido pela Microsoft, o *Windows Image Acquisition*, ou WIA, é o protocolo de aquisição de imagens presente nos produtos Windows, tendo sido introduzido originalmente nas edições Windows Millenium e Windows XP do OS da Microsoft, lançadas em 2000 e 2001, respetivamente.

Enquanto plataforma de aquisição de imagem, o WIA permite a interação entre *softwares* e *hardwares* gráficos, possibilitando a sua comunicação de um modo abrangente e inclusivo, não forçando os *developers*, tanto de *software* como de *hardware*, a ter que adaptar os seus produtos para todas as combinações possíveis entre esses mesmos, servindo então o WIA como uma possível plataforma genérica que qualquer *developer* possa usar. Esta mesma abrangência inclui também uma compatibilidade e uniformização inerente entre todos os *scanners* e aplicações baseados em WIA através de um processo de certificação efetuado pelo próprio Windows. [7]

A primeira versão 1.0 do WIA fornecia suporte para *scanners*, câmaras digitais e outros tipos de equipamentos de vídeo digitais, tendo sido sucedida pela atual versão 2.0, cujo foco passa a ser apenas os *scanners*, deixando de suportar conteúdo de vídeo após a sua introdução no Windows Vista. No entanto, a versão 2.0 suporta ainda, através de um modo de compatibilidade, aplicações *legacy* da versão 1.0 do WIA. [7]

A Microsoft enumera, entre outras, as seguintes funcionalidades [7]:

- Enumeração e visualização dos dispositivos de aquisição de imagem disponíveis;
- Criação de conexões simultâneas a vários dispositivos;
- Aquisição dos dados do dispositivo através de mecanismos de transferência, tanto de performance normal como alta;
- Manutenção das propriedades da imagem através da transferência de dados;
- Notificações sobre o estado do dispositivo e tratamento dos eventos de *scan*;

- Adição da biblioteca de automação do WIA em forma de DLL[1], para a criação de aplicações WIA.

O WIA dispõe ainda de um tratador de eventos de erro, o qual informa o utilizador, duma extensa lista de erros, sobre algum eventual erro encontrado durante o processo. Este erro pode provir tanto de *software* como de *hardware*, incluindo, por exemplo, algum *hardware* não se encontrar ligado, ou a conexão dispositivo-aplicação não estar a conseguir ser alcançada.

2.1.2.2 TWAIN

Lançado em 1992 pelo grupo homónimo (*TWAIN Working Group)*, constituído pelos líderes de mercado da altura - e cujos membros incluem atualmente, por exemplo, a Adobe, a Fujitsu, a Epson ou a Hewlett-Packard - o TWAIN foi a resposta a uma necessidade de mercado para um protocolo de aquisição de imagem que permitisse a comunicação entre dispositivo e aplicação de um modo uniformizado.

O *TWAIN Working Group* destaca os seguintes objetivos e/ou características do TWAIN, entre outros [8][9]:

- Suporte a múltiplas plataformas e dispositivos;
- Compatibilidade retroativa com versões anteriores;
- Capacidade de suporte a vários tipos de dados e ficheiros;
- Extensibilidade e adaptação a tecnologias futuras;
- Fácil implementação;
- Distribuição de um *kit* de desenvolvimento grátis;
- Estabelecimento do TWAIN como padrão na aquisição de imagens.

Tendo sido um dos primeiros do seu tipo, o TWAIN serviu de base para outras aplicações que o precederam, para o qual muito contribuiu a sua construção padronizada. O WIA, por exemplo, numa versão primordial, era baseado numa adaptação do TWAIN, antes de ser concebido o primeiro produto WIA. Como tal, embora o WIA seja exclusivo ao Windows, o TWAIN é também suportado pelo Windows e outros OS, como o Linux e o MacOS.

[1] Biblioteca de funções que são utilizadas por várias aplicações, pelo que se englobam em ficheiros com extensão .dll para uso partilhado, reduzindo a redundância de código

2.1.2.3 *Image and Scanner Interface Specification* – ISIS

Criado em 1990 pela Pixel Translations, o *Image and Scanner Interface Specification*, ou ISIS, foi dos primeiros, se não o primeiro, do seu tipo, tendo sido, tal como o TWAIN, desenvolvido como resposta à necessidade da altura em criar um protocolo uniformizado para *softwares* e *hardwares* de aquisição e tratamento de imagens.

O funcionamento do ISIS é efetuado por um sistema de módulos, no qual cada um desses módulos é encarregue de efetuar uma tarefa específica, como o controlo de *scanners* ou impressoras, a visualização de imagens, a compressão ou conversão de formatos de imagens, ou a leitura e escrita de ficheiros de imagem. [10]

Estes módulos comunicam posteriormente entre si, enviando mensagens com informações contidas em *tags*: valores guardados dentro dos módulos referentes ao estado atual do mesmo, e que podem exprimir características como a resolução, a área de digitalização ou o formato de *output*. Os módulos são então ligados em *pipes* para realizar funções mais robustas. Podemos observar um exemplo abaixo na Figura 2.2:

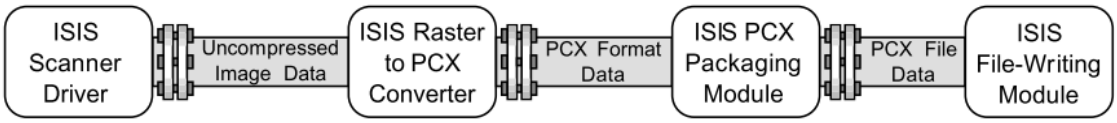


Figura 2.2 – Exemplo de um *pipe* ISIS, onde quatro módulos ligam-se entre si para efetuar uma digitalização, desde a aquisição da imagem digitalizada até à escrita do ficheiro [10]

À semelhança dos protocolos abordados anteriormente, a Pixel Translations fornece também *kits* de desenvolvimento de *software* para a criação de aplicações que estejam em conformidade com o *standard*[2] da ISIS, o AIIM MS61[3], disponíveis para Windows, Mac e Linux. Estes *kits*, ao contrário do observado até agora, não estão disponíveis gratuitamente. No entanto, a Pixel Translations oferece aos seus clientes apoio técnico na compra dos seus produtos, mediante também uma verba anual, havendo também a possibilidade do *developer* ter de pagar *royalties*. [10]

[2] Padrão aceite pelos *developers* como escolha comum entre eles para a criação de *softwares* distintos, possibilitando a interoperabilidade entre eles

[3] Padrão estabelecido em 1996 para a criação de programas para o interface entre *scanners* e programas de *software* de imagem

2.1.2.4 *Scanner Access Now Easy* – SANE

O SANE aparece destacado dos produtos anteriormente analisados, não pela função que pretende cumprir, que é, em concordância com os capítulos anteriores, a de fornecer um protocolo uniformizado para a aquisição e tratamento de imagem, mas sim pelo ambiente em que se insere: a API foi desenvolvida com o objetivo de funcionar em ambientes UNIX, como GNU ou Linux, criando um nicho como alternativa especializada para Linux, em desprezo do TWAIN ou do ISIS. No entanto, se desejado, a API foi desenhada de modo a ser possível de implementar em qualquer sistema operativo. [11]

A implementação do SANE divide-se em dois aspetos distintos: a implementação de um *frontend*, isto é, um programa com interface gráfica que permite a interação do utilizador e que irá enviar pedidos e/ou instruções ao *backend*, ou seja, a *driver*[4] que vai fazer a interface entre API e *hardware*, possibilitando o envio de pedidos de baixo nível de volta ao *frontend*, que irá devolver os dados da imagem digitalizada e convertê-los num formato de ficheiro que o *frontend* consiga lidar e apresentar ao utilizador. [12]

Uma vez que o SANE foi desenvolvido com uma forma uniformizada em mente, só é necessário desenvolver um *backend* por dispositivo, que depois conseguirá fazer interface com o *frontend*. A *driver* do *backend* encontra-se presente no dispositivo em si, se este for compatível com o SANE, uma informação que se encontra disponível no *website* do projecto do SANE. Quanto ao *frontend*, fruto do estado de domínio público do SANE, existem várias opções disponíveis para descarregar, criadas por membros da comunidade de desenvolvimento do SANE. Todas estas estão disponíveis gratuitamente, pelo que o utilizador apenas terá que perceber qual a solução que acha mais adequada, tanto de características como de estética. Também poderá desenvolver a sua própria solução utilizando os recursos disponíveis.

2.1.2.5 *Overview*

Como observado acima, existem várias soluções disponíveis para a aquisição de imagem do *scanner*, tornando-se necessário discernir qual a que melhor se adequa ao projeto.

Imediatamente podemos descartar a implementação do ISIS no projeto: embora pioneira, trata-se de uma solução industrial, disponível maioritariamente para grandes empresas, com necessidades de digitalização com um volume muito maior e um custo insuportável para o de-

[4] Programa associado a um dispositivo, *software* ou *hardware*, que contém funções que permite a comunicação entre o dispositivo e o computador, possibilitando a realização das funções nela presente sem conhecimento prévio detalhado do dispositivo

senvolvimento deste projeto. De seguida, podemos descartar o SANE, visto que a sua utilização principal é em ambientes como o Linux, sendo que este projeto é, nesta etapa, desenvolvido em Windows.

Restando o WIA e o TWAIN, torna-se quase uma questão de preferência entre os dois: o TWAIN, uma vez que antecede o WIA, e não sendo restrito ao Windows, tem uma maior abrangência de *softwares* e de *hardwares* que suporta, o que resulta numa maior comunidade de apoio e também na menor chance de existirem problemas de compatibilidade; o WIA, uma vez que até foi baseado no TWAIN originalmente, tem um funcionamento semelhante, mas criado de forma específica para o Windows, o que resulta num apoio mais especializado dentro do Windows para aplicações WIA, do que no TWAIN para vários ambientes.

Durante a pesquisa para apoio tanto para WIA como para TWAIN, foi realizada a concretização de um tutorial para ambos os protocolos, sendo que o tutorial para WIA foi realizado com sucesso, enquanto que o tutorial para TWAIN não foi possível de concretizar devido a problemas impercetíveis. Com isto em conta, e também devido à semelhança de funcionamento e qualidade das duas APIs, foi escolhido o WIA como protocolo de aquisição de imagem para o desenvolvimento deste projeto.

2.1.3 *Optical Character Recognition* – OCR

Por *optical character recognition* (traduzido como reconhecimento ótico de caracteres), ou OCR, entenda-se como a conversão mecânica e/ou eletrónica de uma qualquer imagem de texto escrito para texto codificado por máquina, tornando-o possível de ser verificado, copiado ou editado.

Este funcionamento é bastante útil para várias áreas, como por exemplo: o reconhecimento de matrículas de automóveis, usado nas operadoras de infraestruturas de transporte, como a Brisa em Portugal; o reconhecimento e preenchimento automático de formulários e impressos em ambientes de hospitais por exemplo, ou a verificação de assinaturas, útil em qualquer validação de identificação, no ambiente bancário por exemplo.

De seguida, serão abordados mais em detalhe alguns OCR's, escolhidos pela sua relevância no mundo dos OCR's:

2.1.3.1 Asprise OCR

Fundada em 1998, a Asprise é uma empresa de desenvolvimento de *software* que se foca em tecnologias de imagens de documentos e de OCR, fornecendo aos seus clientes *kits* de de-

senvolvimento de *software* (ou SDK's) e bibliotecas de programação para a criação de produtos de OCR e de reconhecimento de códigos de barras [13] em várias linguagens de programação, como o C/C++, C#, Java ou o Python [14], sendo possível criar um programa Asprise OCR em vários sistemas operativos, como o Windows, o Linux ou o Mac OS.

A Asprise trabalha com vários parceiros de renome internacional, como o banco Barclays, a Vodafone, a Apple ou até mesmo os Departamentos de Estado e de Segurança Nacional dos Estados Unidos da América, sendo um líder na área do OCR desde a sua criação. [13]

A Asprise apresenta como características dos SDK's [14]:

- o correto e rápido reconhecimento de texto, com parâmetros ajustáveis para favorecer a precisão ou a velocidade;
- funcionamento em mais de 20 línguas, como Inglês, Espanhol, Francês, Alemão ou Português;
- a conversão de ficheiros de imagem como o JPEG, o PNG, o TIFF ou até mesmo o próprio PDF para ficheiros PDF totalmente pesquisáveis;
- ou o apoio aos formatos mais populares dos códigos de barras, entre outros.

O resultado de um programa de demonstração disponível no site oficial da Asprise demonstra a conversão de ficheiro de imagem PNG para um PDF totalmente pesquisável, como é possível ser visualizado abaixo (Figuras 2.3 e 2.4):

Asprise OCR and Barcode Recognition

High performance, royalty-free OCR and barcode recognition on Windows, Linux, Mac OS and Unix.

Asprise OCR (optical character recognition) and barcode recognition SDK offers a high performance library for you to equip your Java applications (Java applets, web applications, Swing/JavaFX components, JEE enterprise applications), C#/VB.NET applications, and C/C++/Python applications with functionality of extracting text and barcode information from scanned documents.

Convert Images To Searchable PDF

With a few lines of code, you can convert various formats of images such as JPEG, PNG, and TIFF into searchable PDF.

PDF Output Formats	Remarks
PDF	Normal PDF
PDF/A	ISO 19005

All Popular Barcode Formats

All popular barcode formats are supported: EAN-8, EAN-13, UPC-A, UPC-E, ISBN-10, ISBN-13, Interleaved 2 of 5, Code 39, Code 128, PDF417, and QR Code.

Figura 2.3 - Imagem de exemplo utilizada na demo do Asprise OCR [15]

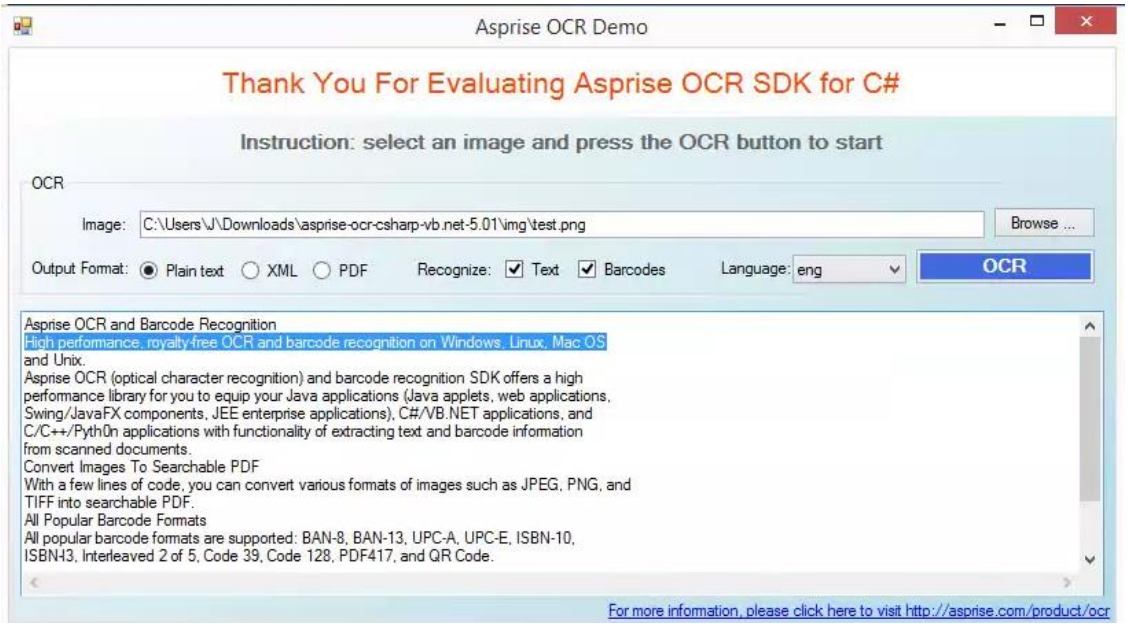


Figura 2.4 – Recorte de um *screenshot* do vídeo da demo do Asprise OCR, onde se pode constatar o correto reconhecimento da imagem da Figura 2.3 [15]

2.1.3.2 ABBYY FineReader

À semelhança da Asprise, a ABBYY é uma empresa de desenvolvimento de software de OCR e de captura de conteúdo através de tecnologias à base de línguas, tanto para computadores [16] (disponível para Windows, Linux e Mac OS) e dispositivos móveis, tendo sido fundada na Rússia em 1989, e conta com parceiros de renome como a Microsoft, a Fujitsu, a Dell, a Panasonic ou a Toshiba.

O seu principal produto é o ABBYY FineReader, uma aplicação de OCR originalmente lançada em 1993 como o primeiro sistema desse género na Rússia, sendo que atualmente, de acordo com dados fornecidos pela própria ABBYY, é utilizado por mais de 30 milhões de pessoas pelo mundo. [16] Como características do FineReader, a ABBYY destaca [17]:

- Conversão precisa (99.8% de precisão [18]) e veloz de documentos em papel digitalizados, imagens e PDF's para ficheiros Word, Excel ou PDF pesquisáveis, entre outros formatos, todos totalmente editáveis e com revisão e correção de texto;
- Fácil edição e gestão de ficheiros PDF's (tanto as imagens digitalizadas como os convertidos), como por exemplo, correção de gralhas e imagens ou anulação de conteúdo dentro de um documento, reorganização de páginas, preenchimento de formulários, etc.;
- Comparação automática de versões distintas de um mesmo documento, inclusive entre formatos diferentes, como por exemplo entre um ficheiro PDF e um ficheiro Word;
- Reconhecimento de 192 línguas diferentes, com 48 a usufruir de apoio de um dicionário inerente à aplicação, sendo que 35 destas podem ser comparadas entre si.

Nas Figuras 2.5 e 2.6 abaixo, podem ser visualizados dois exemplos do funcionamento do ABBYY FineReader:

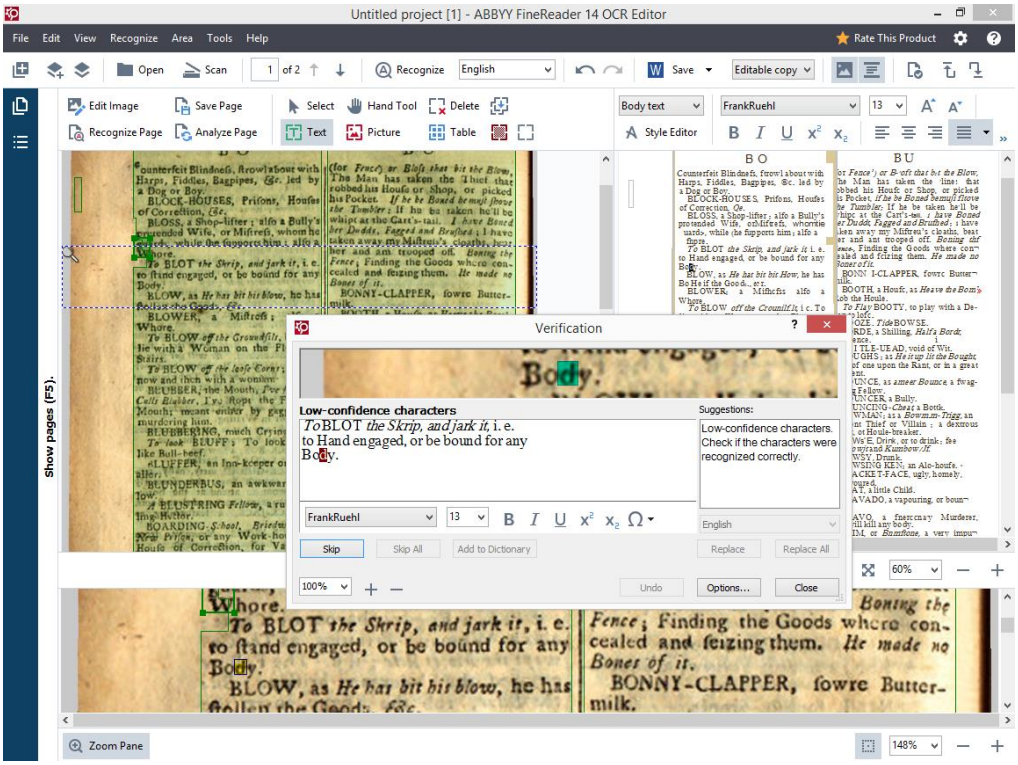


Figura 2.5 - Conversão e verificação de um texto digitalizado no FineReader [19]

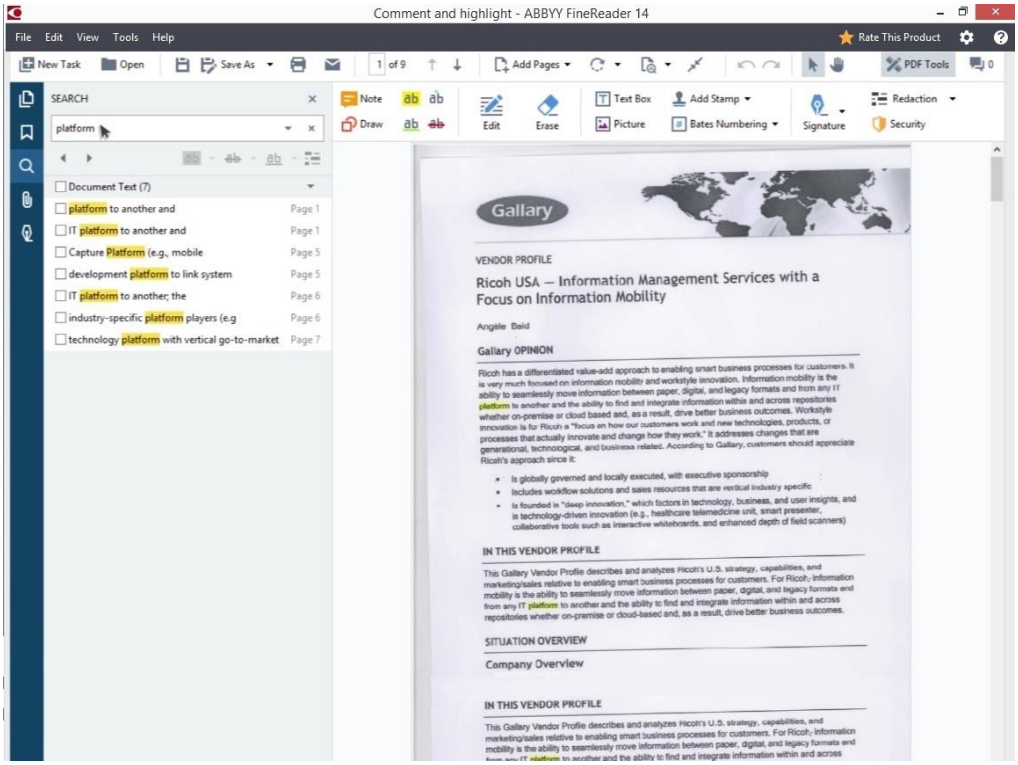


Figura 2.6 - Pesquisa num ficheiro PDF proveniente de uma digitalização [19]

2.1.3.3 Tesseract

Desenvolvido nos laboratórios da Hewlett-Packard entre 1984 e 1994, o Tesseract é um motor para a criação de programas OCR, tendo sido escrito em C e C++, estando disponível para desenvolvimento em Linux, Windows e, de um modo não oficial, para Mac OS. [20]

O Tesseract destaca-se dos OCR's anteriormente descritos por ser *open-source*, ou seja, do domínio público, sendo que qualquer um pode utilizar o Tesseract para criar o seu próprio programa ou aplicação de OCR e distribuir pelo público. Foi lançado nesse âmbito em 2005, sendo que a Google patrocina o desenvolvimento de aplicações através do Tesseract. [21]

Outra diferença é também o facto de que o Tesseract é um OCR bastante *bare-bones*: isto é, não possui uma *Graphical User Interface* (interface gráfica do utilizador), ou GUI, por natureza, sendo todo o processo realizado através da linha de comando; a imagem de fonte tem também de ser pré-processada de modo a poder ser analisada pelo Tesseract, o que inclui operações de processamento de imagem como o escalamento ou a binarização da imagem, por exemplo. [22]

No entanto, o Tesseract é elogiado pela sua precisão no reconhecimento de caracteres dentro dos OCR's *open-source*, que pode ser melhorada adicionalmente ao treinar o programa com um conjunto de treino, tal como as línguas suportadas, as quais se contam mais de 100 inicialmente. Como ficheiros de saída, o Tesseract consegue produzir ficheiros TXT, PDF ou até mesmo HTML, sendo o TXT o tipo de ficheiro por omissão, e o PDF possível de ser pesquisável. [23]

2.1.3.4 Overview

Analisados os três OCR's descritos acima, podem-se tirar algumas conclusões ao compará-los; logo à cabeça, o Tesseract é claramente o vencedor na vertente do preço, sendo que se trata de um *software* grátis e aberto para toda a comunidade, ao contrário do FineReader da ABBYY, que pratica preços a partir dos 200 até aos 500 euros, sendo que o utilizador garante acesso a mais funcionalidades quanto mais pagar. Já o Asprise OCR pratica preços base a partir dos 5000 dólares – valores insustentáveis para alguém que pudesse eventualmente considerar o seu uso na realização de uma tese de mestrado como esta.

Em termos de qualidade, tanto a Asprise como a ABBYY gabam-se de terem reconhecimento de caracteres de topo, como se pode ver pelos exemplos demonstrados nos seus *sites* bem como pelo seu estatuto como empresas de renome mundial na área. O Tesseract peca comparativamente neste aspeto, pois precisa de ser treinado para melhorar e as imagens que

são processadas precisam de um pré-processamento – no entanto, esses dois fatores podem conduzir a uma substancial melhoria da qualidade dos resultados provenientes do Tesseract.

Sendo que dos três apenas o OCR da ABBYY possui um GUI, é o FineReader que vence a disputa da apresentação, possuindo uma interface como vista acima na Figura 2.5, na qual se consegue observar várias janelas de relevo durante o processo de reconhecimento.

No final, o FineReader demonstra ser, até prova em contrário, o melhor entre os três: é o que vem, à partida, com melhor qualidade no reconhecimento e em todas as vertentes associadas, e com maior facilidade de uso, sendo que se trata da simples instalação de um software. No entanto, sofre um grande revés no custo, fator também desde logo decisivamente eliminatório para com o OCR da Asprise – preço base de 5000 dólares será para vender como produto para empresas – o que torna o Tesseract como principal eventual candidato a ser trabalhado numa tese de mestrado: grátis, apoiado e desenvolvido em comunidade e apresenta resultados com qualidade, os quais ainda conseguem ser melhorados ao trabalhar o pré-processamento das imagens a processar.

2.1.4 FINA

2.1.4.1 Especificações

Produzido pela Dreampath Diagnostics, empresa francesa sediada em Estrasburgo, existe o FINA: a primeira solução totalmente automática para a gestão de blocos de parafina, construída para a diminuição de erros no laboratório, perdas de blocos de parafina e de atrasos nos diagnósticos aos pacientes. [24]

Figura 2.7 - *Scanner* FINA

O FINA funciona como um dispositivo *scanner*, o qual possui uma ranhura na qual é possível inserir gavetas com blocos de parafina no seu interior, com a finalidade de digitalizar os seus códigos de identificação, registando também as suas posições relativas dentro da gaveta e a identificação da gaveta em si, possibilitando a fácil procura dos blocos na sua sala de armazenamento.

Não é, no entanto, vendido apenas o FINA em si – com ele é fornecido em pacote os seguintes produtos [25], que podem ser visualizados mais abaixo na Figura 2.8:

- As gavetas personalizadas de armazenamento dos blocos que funcionam com o FINA;
- Um módulo arquivador de gavetas;
- Um PC especializado com o *software* criado para a gestão do armazenamento dos blocos;
- Um dispositivo portátil de mão ao estilo de um PDA para a leitura e consulta de blocos à distância;
- E ainda uma impressora de etiquetas para codificação das gavetas.

Figura 2.8 - Componentes vendidos pela Dreampath na solução FINA: a) gavetas personalizadas para uso com o FINA, b) arquivador de gavetas, c) ecrã de entrada do *software* de gestão do armazenamento de blocos, d) dispositivo portátil de leitura de blocos. Impressora de etiquetas não visualizada

2.1.4.2 Funcionalidades e Características

Com a utilização de um FINA, uma equipa médica num laboratório de patologia consegue [24][25]:

- Assegurar a localização dos blocos através do *software* de gestão, permitindo ao utilizador pesquisar por bloco dentro de uma base de dados;
- Automatizar e simplificar o processo de armazenamento e gestão dos blocos, permitindo ao utilizador verificar as datas de requisição e de entrega de um qualquer bloco, incluindo a sua razão de requisição ou o requisitante do bloco;
- Melhorar a segurança dos blocos e, consequentemente, dos seus pacientes, ao eliminar os erros nas etapas de armazenamento;
- Acelerar os diagnósticos e com melhor precisão, dispondo de 240 espaços para blocos em cada gaveta, que não precisam de estar em nenhuma ordem específica, sendo que cada gaveta é completamente digitalizada em 2 a 3 minutos;
- Permite a integração de um FINA num sistema informático de laboratório pré-existente.

2.1.4.3 Overview

No fundo, o *scanner* FINA seria o objetivo final ideal na realização desta tese: todos os objetivos propostos anteriormente são cumpridos pelo funcionamento do FINA, e inclui ainda outros produtos como o dispositivo para a consulta portátil dos blocos ou a impressora de etiquetas para gavetas.

No entanto, o facto da solução apenas ser vendida como pacote de vários produtos, é pouco atraente para um possível consumidor: um laboratório patológico pode apenas desejar adquirir o *scanner* e o software, de modo a ter uma base de dados informática mais fiável e menos suscetível ao erro humano – terá, mesmo assim, de adquirir, por exemplo, as gavetas personalizadas, que são as únicas que funcionam com o FINA, e ainda os arquivos de gavetas para as armazenar, tudo isto acarretando custos adicionais que, muito provavelmente, serão insuportáveis.

No caso específico do IVT, por exemplo, as gavetas utilizadas em nada têm a ver com as gavetas personalizadas do FINA: enquanto as gavetas do IVT são compridas e estreitas, as do FINA são mais semelhantes a bandejas, possuindo uma maior área de contacto com a superfície de digitalização por gaveta. Caso o IVT desejasse adquirir um produto FINA, seria obrigado a concretizar uma extensa remodelação do seu sistema organizacional, o que é altamente indesejável.

Em suma, a solução FINA poderá servir como base para a construção desta tese, com a ressalva de ser menos restritiva e mais barata que o FINA, não obrigando à aquisição de novos recursos como gavetas, mas procurar utilizar as gavetas utilizadas pelo IVT, cujos custos serão mínimos, visto que se trata de uma tese realizada em parceria com o mesmo IVT.

3
Fundamentos Teóricos

Neste capítulo, serão apresentados ao leitor os fundamentos teóricos necessários para compreender como foi construído o *software* principal da tese, as mecânicas que ele engloba, como é executado o seu funcionamento, e a lógica por trás dos algoritmos existentes no programa.

Em primeiro lugar, é exposto ao leitor uma sucinta introdução ao conceito de processamento de imagem: o que é, como se faz ou para que serve são algumas questões que se pretende responder. De seguida, é explicado o funcionamento e processo de OCR, bem como a maneira que este e a questão de processamento de imagem se interligam, tanto um com o outro, como com este projeto. Finalmente, é aprofundado o algoritmo desenvolvido para a realização do processo de OCR no *software* principal, cujo funcionamento base foi inspirado em *papers* pré-existentes e adaptado para o contexto deste problema.

3.1 Processamento de Imagem

3.1.1 Imagem Digital

Antes de tentar perceber o conceito de processamento de imagem, é primeiro necessário entender o conceito de imagem. Muitos entenderão uma imagem como uma qualquer representação visual de um qualquer conceito visível, sendo que esta representação pode ser uma memória na mente de uma pessoa, ou impressa em papel ou num ecrã digital. Enquanto que, para visualizar uma imagem mental, o ser humano não precisa de outro conceito, um computa-

dor, por exemplo, precisa de entender o conceito de imagem de um modo que o permita imprimi-la.

Uma imagem pode, então, ser considerada como uma função f(x,y), em que x e y são coordenadas espaciais, geralmente abcissa e ordenada, em que, para cada ponto de coordenadas (x,y), corresponde-lhe um valor f de intensidade. Quando os valores de f, x e y forem todos valores finitos e discretos[5], podemos assumir a imagem como sendo uma imagem digital, possível de ser processada digitalmente. [26]

A cada ponto f(x,y) de uma imagem digital, é-lhe dado o nome de *pixel*, sendo este o elemento básico de uma imagem. Um *pixel* pode, geralmente, assumir um valor f de intensidade que pode ir do valor 0 ao 255, sendo que 0 corresponde ao preto, 255 corresponde ao branco, e todos os valores intermédios são tonalidades de cinzento entre esses dois valores.

Para esclarecer melhor o que foi referido acima, foi construída uma imagem, de 6x6 *pixels*, centrada num referencial cartesiano. Podemos tomar os seus 6x6 *pixels* como sendo as possíveis coordenadas da imagem, sendo que a cada ponto irá corresponder um certo f(x,y). Para valores f(0,y), por exemplo, teremos f = 0, pelo que o *pixel* terá uma tonalidade preta. A cada valor que se aumenta em x, o valor de f irá aumentar, até f(5,y), onde f = 255, concedendo ao *pixel* uma tonalidade branca.

Podemos ver abaixo na Figura 3.1 a referida ilustração:

[5] Que disfruta de um número finito de valores contáveis entre dois possíveis valores.

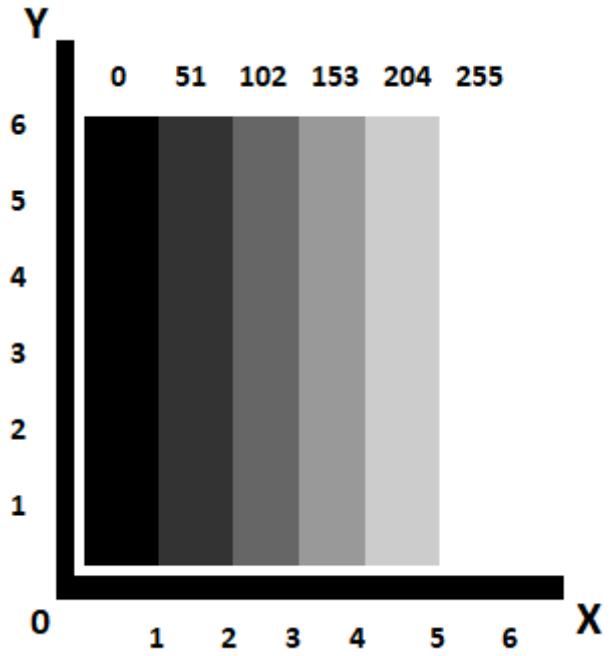


Figura 3.1 – Ilustração de uma imagem de tamanho 6x6 *pixels*, em que a cada valor de X que adiciona, todos os *pixels* dessa coluna têm um valor de intensidade diferente

3.1.2 Processamento Digital

Estabelecido o conceito de imagem, segue logicamente que se procure entender o conceito de processamento: tal como o nome indica, para haver processamento de uma imagem, é-lhe efetuada um certo processo. Este pode ser classificado de três maneiras diferentes: como sendo de baixo nível, de nível médio ou de alto nível. [26]

Um processo de baixo nível é geralmente constituído por operações mais básicas, geralmente efetuadas em questões de pré-processamento de uma imagem, como a redução de ruído, o enriquecimento de contrastes ou o realce de contornos. Todas estas operações são facilmente efetuadas pela aplicação de um filtro à imagem, o que é característico de um processo de baixo nível, no sentido em que tanto o *input* como o *output* do processo são imagens. [26] Na Figura 3.2 abaixo, podemos observar o efeito de um filtro de média aplicado a uma imagem:

Figura 3.2 – Imagem de um olho nas seguintes condições: a) sem qualquer filtro, b) com um filtro de média 3x3, c) com um filtro de média 5x5 [27]

Um processo de nível médio já implica a realização de operações mais complexas, baseadas na conjunção de vários processos de baixo nível, como a segmentação de uma imagem, ou a classificação de objetos numa imagem. Neste tipo de processos, o *output* deixa de ser uma imagem, e passa a ser um atributo ou uma característica, extraídos da imagem e que permitam identificar uma região, uma forma ou um objeto na imagem. [26] É possível observar abaixo na Figura 3.3 um exemplo de características possíveis de serem extraídas através dum processo de nível médio:

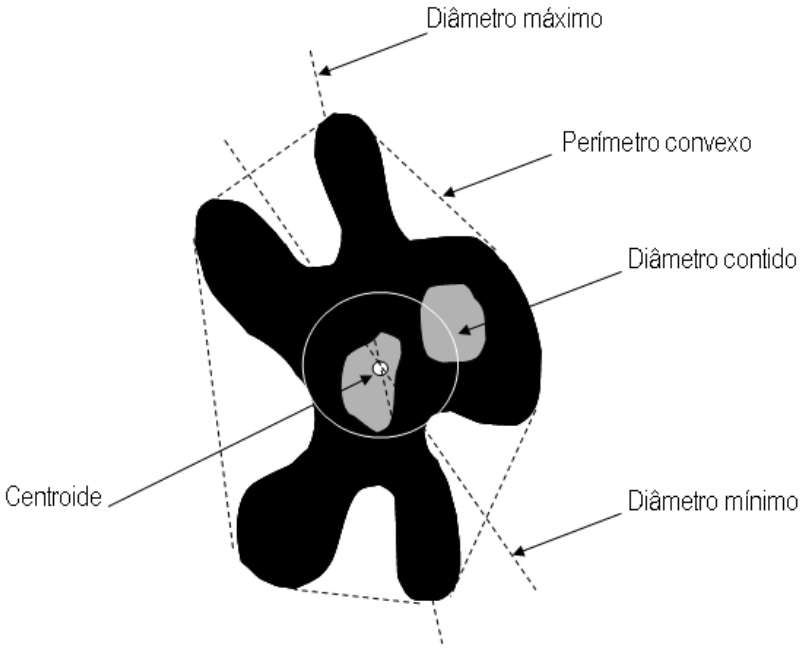


Figura 3.3 - Múltiplas características possíveis de serem extraídas duma região [28]

Finalmente, um processo de alto nível será uma tentativa o mais parecida possível a uma classificação efetuada pela visão de uma pessoa: tal e qual como os nossos olhos vêm uma cadeira e imediatamente pensarão "cadeira", um processo de alto nível visa alcançar essa mesma capacidade de deteção e classificação. Essencialmente, a complexidade do processo e das suas operações inerentes aumenta numa proporção direta para com o nível do processo.

Como poderemos observar de seguida, o processamento digital é o alicerce para a fundação de toda a mecânica do processo de OCR.

3.2 Optical Character Recognition - OCR

O funcionamento geral de um algoritmo OCR pode ser demonstrado pela Figura 3.4 abaixo [29]:

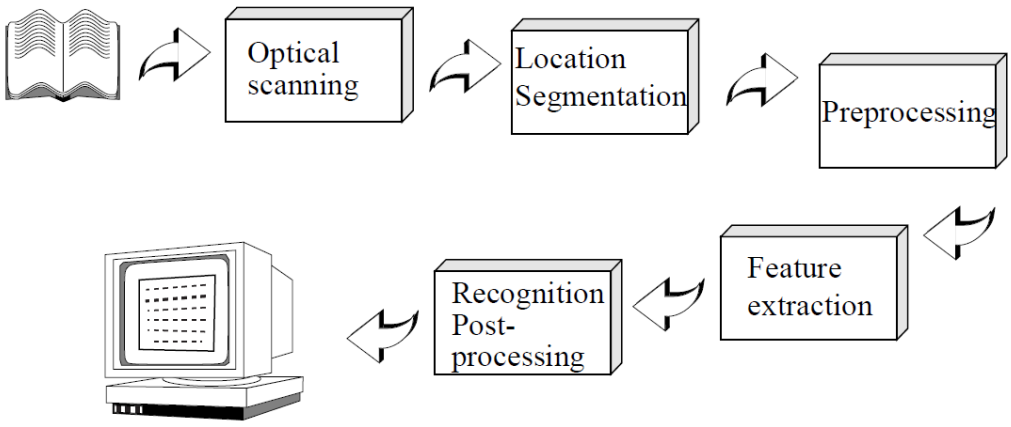


Figura 3.4 - Etapas comuns num sistema OCR

De um modo básico, podem-se explicar as etapas do seguinte modo [29]:

1. Digitalização ótica – onde a imagem a processar é capturada pelo *scanner* utilizado pelo OCR;
2. Localização e segmentação – onde se isolam os componentes desejados pelo utilizador do restante "lixo" indesejado;
3. Pré-processamento – no qual se reduz o ruído inerente à recriação digital da imagem, na tentativa de o caracter ser reconhecido o melhor possível. Pode haver uma certa fluidez de processo entre esta etapa e a anterior, sendo que tanto uma como outra podem anteceder a restante;
4. Extração de características – onde se realiza, efetivamente, a análise do caracter em si, a qual pode ser feita, por exemplo, através de comparações pixel-a-pixel, por simetria, a contagem de círculos fechados, entre outros;
5. Classificação – onde se identifica e atribui ao caracter qual a sua classe de caracter, ou seja, qual caracter é;
6. Pós-processamento – onde se agrupam os caracteres identificados e se corrigem eventuais erros no seu reconhecimento, tendo em conta informações previamente adquiridas pelo sistema. Estas informações podem vir na forma de palavras, dicionários ou conjunções populares de palavras inseridas no sistema, ou através do treino do mesmo: o OCR é testado várias vezes com um conjunto de palavras e caracteres de treino, de modo a que este aprenda que caracteres podem vir a seguir a quais, por exemplo, o que lhe irá conferir uma maior inteligência na altura do reconhecimento dos caracteres.

3.2.1 Digitalização Ótica

É nesta primeira etapa do processo de OCR em que se capta a imagem a analisar: utilizando um *scanner*, este converte a intensidade da luz presente no documento através de um sensor para vários níveis de cinzento. No caso de documentos impressos, no entanto, estes são geralmente escritos a letra preta em papel branco, o que torna possível a conversão da intensidade da luz não em vários níveis de cinzento, mas sim numa nomenclatura binária de preto ou branco: este é o chamado processo de *thresholding*, onde se substitui o valor de intensidade de um *pixel* para preto ou branco, dependendo se este for maior ou menor que um certo valor de *threshold*[6]. [29]

O processo de *thresholding* assume duas formas de o concretizar: através de um valor de *threshold* global, para todo o documento; ou através de um *thresholding* adaptativo, onde o valor de *threshold* se adapta às várias zonas do documento, dependendo do contraste existente.

Ao permitir ao utilizador a criação de imagens binarizadas, este processo permite conseguir um melhor nível de qualidade e fiabilidade no reconhecimento que se segue. Abaixo segue a Figura 3.5, onde se demonstra o processo de *thresholding*.

[6] Limite.

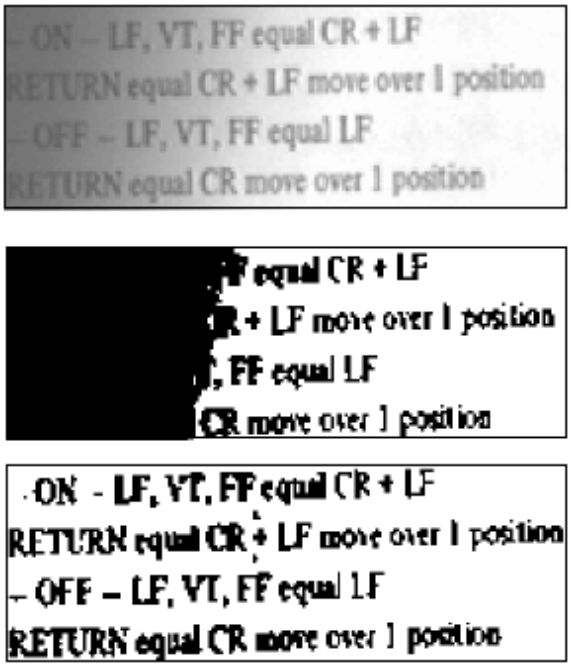


Figura 3.5 - Processo de *thresholding* em 3 tempos: a) imagem digitalizada, b) imagem binarizada através de um *threshold* global e c) imagem binarizada através de um *threshold* adaptativo [29]

3.2.2 Localização e Segmentação

De modo a obter apenas a informação que interessa ao utilizador, é necessário saber conseguir localizar a informação, e segmentá-la, ou seja, separar a informação que realmente importa do resto da imagem, que, para todos os efeitos, será equivalente a lixo ou ruído.

Em termos de reconhecimento, a segmentação é o processo empregue para isolar caracteres e palavras dentro de um texto. Este isolamento é conseguido, primeiro ao encontrar o texto, separando-o palavra e palavra, e essas separadas, letra a letra. Este resultado pode ser obtido através de técnicas específicas de segmentação, como por exemplo, o método dos componentes ligados, onde a cada *pixel* preto se atribui um valor de "etiqueta", as quais se vão propagando aos outros *pixels* pretos, ligando assim os componentes que estiverem conectados.

No entanto, de modo a minimizar os erros nesta fase, que é de uma extrema importância, há que ter cuidado com os seguintes problemas recorrentes: a extração de caracteres fragmentados ou ligados, a distinção entre ruído e informação e a distinção entre texto e figuras. [29] Qualquer equívoco durante a realização desta etapa pode comprometer seriamente os resultados obtidos.

3.2.3 Pré-processamento

Após a sua digitalização, a imagem digital pode ter ruído na sua constituição – este pode ter originado no documento impresso, se esta for de má qualidade, ou na digitalização, se o *scanner* tiver uma fraca resolução de digitalização. Este ruído pode ser um entrave ao reconhecimento bem-sucedido, no entanto, pode ser tratado e anulado ao se realizar um pré-processamento à imagem.

Uma forma comum de ruído é a existência de *pixels* pretos dispersos pela imagem onde não deveriam existir. A solução passa pela aplicação de um filtro de média: este irá aplicar uma função de média a todos os *pixels*, o que, tendo em conta a natureza mais dispersa do ruído em comparação com os *pixels* constituintes da imagem, fará com que estes sejam retirados e substituídos pelo valor médio da intensidade dos *pixels* que os rodeiam.

Outra preocupação a ter nesta etapa é a normalização do texto: isto inclui o tamanho e a inclinação dos caracteres – se o caracter for muito pequeno, pode ser complicado de ser reconhecido com sucesso, e se este tiver até mesmo uma ligeira inclinação, pode ser motivo suficiente para o reconhecimento já não ser sucedido.

É possível observar abaixo, na Figura 3.6 uma ilustração de um pequeno pré-processamento a apenas um caracter:

Figura 3.6 - Normalização da orientação e *smoothing* de um caracter [29]

3.2.4 Extração de Características

Nesta etapa do reconhecimento, o objetivo é averiguar características significativas nos caracteres segmentados que possam auxiliar o seu reconhecimento. Existem várias técnicas que se podem utilizar, e outras tantas características que podem ser extraídas e analisadas num caracter: podem-se verificar medidas como o diâmetro, a área ou o perímetro; podem-se fazer comparações pixel-a-pixel, por simetria; pode-se fazer a contagem de círculos fechados num caracter, entre muitas outras maneiras.

A Figura 3.7 abaixo ilustra um exemplo de investigação às simetrias de 3 caracteres diferentes:

Figura 3.7 - Estudo de simetrias em 3 caracteres diferentes

Como é possível observar na Figura, o caracter "1" não pode ser identificado pela sua simetria, visto que não é simétrico, logo não possui eixos de simetria. O caracter "3" é simétrico na horizontal, possuindo um eixo de simetria, o que lhe permite ser identificado por isso. Já o caracter "8" é simétrico tanto na horizontal, como na vertical. Isto concede-lhe dois eixos de simetria, o que facilita o seu reconhecimento, visto que são poucos os caracteres com esta característica.

3.2.5 Classificação

É nesta etapa do processo de reconhecimento em que se atribui a cada caracter a sua classificação, mediante toda a informação para trás obtida e características extraídas.

No entanto, mesmo com todas as etapas para trás, aqui ainda existe a possibilidade de haverem erros, no caso de caracteres semelhantes: um "1", um "I" maiúsculo e um "L" minúsculo são três caracteres muito semelhantes, pelo que será necessário saber distinguir letras de números na transição de caracteres de volta para palavras.

De modo a atenuar o impacto que essas confusões possam vir a ter no resultado final do reconhecimento, podem-se utilizar técnicas de classificação mais robustas, como a implementação modelos estatísticos ou de redes neuronais no classificador, o que poderá evitar estes erros.

3.2.6 Pós-processamento

Finalizando o processo de reconhecimento, é feito um pós-processamento à informação obtida: os caracteres soltos serão de novo reagrupados, constituindo palavras cujo conteúdo será agora conhecido. Incide também nesta etapa a deteção de possíveis erros e a sua correção: esta pode ser realizada pela inserção de contexto ou de sintaxe na palavra, ou até mesmo pelo uso de um dicionário inserido no sistema com palavras existentes que o processo poderá consultar.

Outra possível via para a correção de erros passa pelo treino do sistema de reconhecimento, testando-o várias vezes com conjuntos de palavras e caracteres de treino, de modo a que o sistema ganhe um conhecimento probabilístico do seguimento provável de caracteres, podendo prever com maior precisão os caracteres que podem anteceder e proceder outros caracteres.

3.3 *Algoritmo de Localização*

Na sequência do capítulo anterior, serve este para explicar ao leitor de um modo mais detalhado o algoritmo específico utilizado para o desenvolvimento desta tese.

O algoritmo desenvolvido em questão foi criado no seguimento da conclusão da unidade curricular de Sistemas Sensoriais; mais em concreto, na conclusão do projeto final da componente prática, onde foi proposto aos alunos a criação de um algoritmo de deteção e reconhecimento de matrículas em imagens nas quais se podiam observar diversos tipos de automóveis, dispostos também em variadas maneiras diferentes.

De modo a auxiliar os alunos nesta tarefa, foram disponibilizados dois *papers* para consulta [30][31], a partir dos quais se adaptaram as características mais apetecíveis para a criação de um algoritmo para este projeto em particular.

Como estabelecido no capítulo 2.1.1, será adquirido um *scanner* de mesa, de modo a poder digitalizar as gavetas facultadas pelo IVT. Tanto o *scanner* como as gavetas ambos possuem uma forma retangular, sendo que as gavetas são estreitas, enquanto que o scanner é comparativamente mais largo.

Tem-se então que o resultado de uma digitalização será uma (ou várias) gaveta/s dispostas paralelamente entre si e em relação à janela de digitalização em que se inserem. As gavetas são de cor cinzenta clara, enquanto que a base do suporte de madeira tem uma cor bege clara. Isto vai provocar um contraste na imagem, o qual pode ser utilizado pelo algoritmo para encontrar as gavetas dentro da imagem.

3.3.1 Binarização Otsu

Para utilizar o contraste existente na imagem, embora seja possível processá-la de raiz, é uma prática melhor efetuar um pré-processamento em primeiro lugar. O objetivo passa por, como mencionado no capítulo 3.2.1, realizar um processo de *thresholding*, de modo a obter uma imagem a tons apenas de preto e branco, o que irá facilitar bastante a procura de limites.

Uma das maneiras possíveis de se fazer um processo de *thresholding*, e a utilizada no algoritmo desenvolvido, é o método de Otsu: criado pelo investigador homónimo Nobuyuki Otsu, trata-se de um método que, mediante um histograma bimodal, procura calcular o valor de *threshold* que minimize as variâncias dos valores intra-grupos, obtendo-se um nível de tons escuro e claro o mais homogéneos possível. [32]

Um histograma refere-se a um gráfico que demonstra a quantidade de níveis de intensidade de cinzentos presentes numa imagem – numa imagem com 256 níveis de cinzento, por exemplo, o histograma será composto por 256 valores em X, cada um com o seu valor de intensidade Y correspondente, como se pode observar na Figura 3.8 abaixo:

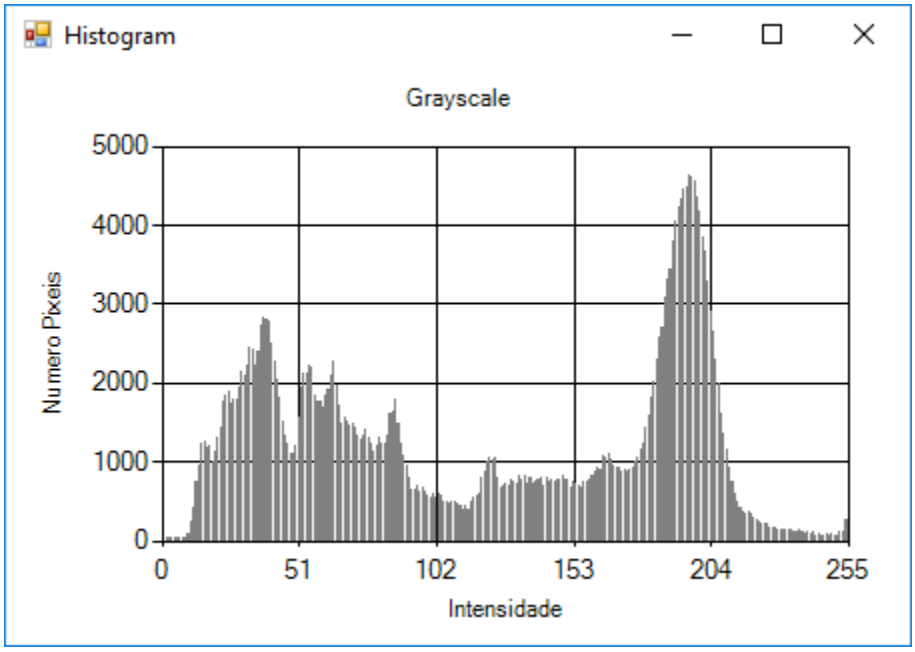


Figura 3.8 – Histograma dos níveis de cinzento de uma imagem

Analisando a Figura 3.8, podemos observar uma grande prominência de *pixels* com valores de intensidade entre os 175 e os 200, e uma segunda prominência, menos destacada, à volta do valor de intensidade de 50, notando-se uma clara tendência para a separação entre os pontos claros e os pontos escuros. Um histograma diz-se bimodal se essa separação estiver bem definida, dividindo os pontos entre dois grupos, como representado abaixo na Figura 3.9:

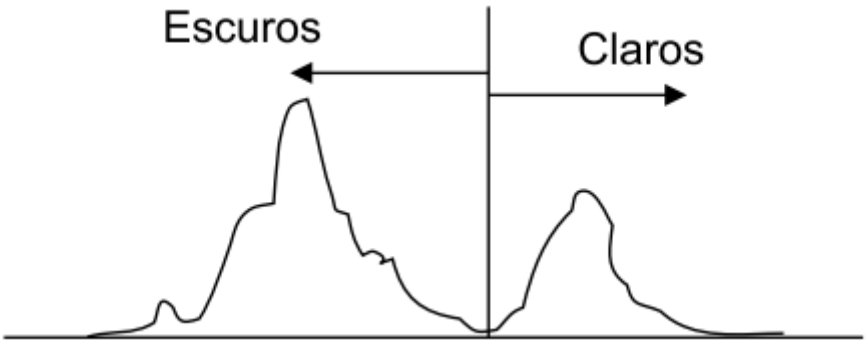


Figura 3.9 - Histograma Bimodal [32]

Tendo um histograma bimodal, e assumindo dois grupos 1 e 2, pode-se então tentar minimizar o valor da variância intra-grupos através de cálculos, aplicando as seguintes fórmulas:

$$q_1(t) = \sum_{i=1}^{t} P(i) \tag{1}$$

$$q_2(t) = \sum_{i=t+1}^{I} P(i) \tag{2}$$

$$\mu_1(t) = \frac{\sum_{i=1}^{t} iP(i)}{q_1(t)} \tag{3}$$

$$\mu_2(t) = \frac{\sum_{i=t+1}^{I} iP(i)}{q_2(t)} \tag{4}$$

$$\sigma_1^2(t) = \frac{\sum_{i=1}^{t} (\mu_1(t) - i)^2 \cdot P(i)}{t} \tag{5}$$

$$\sigma_2^2(t) = \frac{\sum_{i=t+1}^{I} (\mu_2(t) - i)^2 \cdot P(i)}{(I - t)} \tag{6}$$

$$\sigma_W^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t) \quad (7)$$

As equações (1) e (2) são referentes ao cálculo da probabilidade de um dado *pixel* ser do grupo 1 ou 2, em função de um certo valor t; as equações (3) e (4) são referentes ao cálculo do valor médio de cada grupo; as equações (5) e (6) são referentes ao cálculo da variância de cada grupo e a equação (7) é referente ao cálculo da variância conjunta dos dois grupos.

O objetivo passa por calcular (7) para cada um dos valores de intensidade t = [0, 255], de modo a descobrir qual o valor t que minimiza o resultado de (7). Atendendo ao volume de operações a realizar, todo o cálculo é feito de um modo computacional e automático, não só aumentando a velocidade de resolução, bem como diminuindo o erro humano associado.

3.3.2 Localização e Segmentação

Uma das técnicas de localização e segmentação utilizadas em [30], [31], [33] consiste na deteção de limites ao procurar áreas de contraste elevado: no caso das matrículas, por exemplo, estas apresentam um contraste bastante definido e facilmente observado, havendo transições de cor na zona do país para os caracteres da matrícula (azul para branco), entre os caracteres e o fundo em que estes se encontram (branco para preto e vice-versa) e entre os caracteres da matrícula para a zona do ano e mês (branco para amarelo). Estas transições podem ser facilmente obtidas e representadas graficamente ao calcular a primeira derivada da intensidade da cor ao longo de uma linha na imagem, resultando em picos referentes à localização provável da matrícula. Podemos observar esta ocorrência nas figuras 3.10 e 3.11 abaixo:

Figura 3.10 - Imagem de um carro, com duas linhas destacadas para analisar a sua 1ª derivada [30]

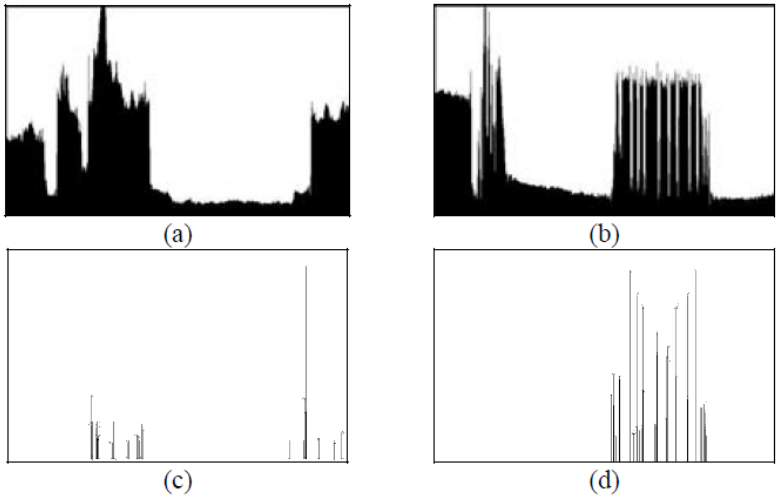


Figura 3.11 – Valores de intensidade da cor para: (a) a linha A, (b) a linha B; 1ª derivada da intensidade da cor ao longo da (a) linha A e da (b) linha B (sem as variações menos significantes) [30]

Uma vez que a linha B é, entre as duas, a única que coincide com uma linha interior à matrícula, esta apresenta vários picos seguidos, numa gama de valores de coordenada que coincidem com a provável localização horizontal da matrícula. Analogamente, o mesmo processo pode ser utilizado para averiguar a provável localização vertical da matrícula, e, ao juntar os resultados, obter-se-á uma localização provável da matrícula em si, cabendo depois ao programador a imposição de condições extra a investigar para a diminuição da localização provável e consequente verificação exata dos limites da matrícula.

3.3.3 Aplicação do Algoritmo

Ao perceber o conteúdo explicado acima nos capítulos 3.3.1 e 3.3.2, tornou-se possível a adaptação desses conhecimentos para a construção de um algoritmo de localização, que será aplicado na situação descrita no início do capítulo 3.3, nomeadamente a/s gaveta/s disposta/s na área útil de digitalização do *scanner*.

Na área de digitalização, o *scanner* apenas poderá visualizar duas coisas: a base do suporte, ou gavetas. Estes dois intervenientes não só têm cores distintas entre si, como também a fonte de luz proveniente do *scanner* é aplicada de igual modo em toda a superfície de digitalização - pelo que se notará um contraste elevado entre base e gaveta/s.

Segue então que, aplicando-se um processo de binarização de Otsu à imagem digitalizada, facilmente se conseguirá distinguir entre base e gaveta/s: como veremos mais à frente, a imagem tomará um fundo preto onde quer que esta encontre a base do suporte, com *pixels* brancos impostos em cima desse fundo onde quer que se encontrem gavetas. Partindo daí, através da variância de intensidades, torna-se possível localizar as gavetas.

Localizadas as gavetas, e continuando a utilizar o método das variâncias, tornar-se-á possível segmentar a gaveta em cada vez mais componentes interiores: da gaveta partimos para a localização individual de cada bloco, o qual podemos dividir em duas metades, superior – a qual se pode dividir também em duas metades esquerda e direita – e inferior – que se pode dividir em três partes.

O programa, ao conseguir localizar e segmentar todos estes campos, permite a realização de um reconhecimento muito mais específico e seguro dos caracteres inscritos nos blocos, possibilitando o posterior tratamento e uso dos dados recolhidos da parte do/s utilizador/es.

4

Metodologia e Implementação

Neste capítulo serão expostos o caminho e as etapas percorridas para a construção e implementação da proposta de tese: a criação do protótipo da caixa envolvente; a escolha e integração de *softwares* adicionais para auxiliar o funcionamento do *software* principal, e o desenvolvimento do *software* principal – numa primeira instância, será exposta uma vista geral ao funcionamento do *software*, às suas funcionalidades e ao contexto em que elas podem ser chamadas. De seguida, será exposta uma vista mais aprofundada ao *software*, abordando as classes constituintes do programa e as funções que o permitem correr da forma pretendida, bem como apresentando ao leitor vários fluxogramas que retratam o caminho percorrido com maior clareza sobre alguns dos processos efetuados.

4.1 Protótipo

4.1.1 Scanner

Como mencionado anteriormente no capítulo 2.1.1, foram averiguadas as condições de aquisição de um *scanner* de mesa, cujo papel será o de elemento digitalizador na conceção do protótipo do sistema de digitalização de blocos.

A escolha acabou por recair no *scanner* EPSON Perfection V39, sendo que este reuniu as condições necessárias para seguir em frente com o projeto: trata-se de um modelo *flatbed*, com tampa amovível, o que facilita a sua adaptação na criação do protótipo, e teve um custo suportável para a sua compra. Em termos estéticos, é praticamente idêntico ao modelo Perfection V19, ilustrado na Figura 2.1.

4.1.2 Caixa Envolvente

Devido à sua composição natural, o *scanner* de mesa adquirido suporta naturalmente a digitalização de documentos com a sua superfície de digitalização orientada para cima. No entanto, a digitalização dos blocos histológicos pressupõe que estes sejam também orientados para cima, o que implica a obrigatoriedade de se orientar a superfície de digitalização para baixo.

Naturalmente, simplesmente colocar o *scanner* por cima da gaveta com os blocos não é uma solução atraente, tendo tanto de precariedade, visto que a gaveta não tem a dimensão suficiente para ser utilizada como suporte, como de falta de elegância de engenharia. Desse modo, desenhou-se a construção de uma base de suporte para o *scanner*, dentro da qual haveria espaço para inserir confortavelmente as gavetas a digitalizar.

A construção da base foi, não obstante a pequena aventura à esfera da bricolage, um processo simples de efetuar, até porque o desenho trata-se de uma construção simples: de modo a assemelhar-se a uma caixa envolvente, e tendo em conta o desenho paralelepipédico do *scanner*, foi idealizada a construção de uma base com duas paredes montadas em cima da mesma, sendo que o *scanner* seria colocado em cima das paredes. Esta construção foi realizada com placas de madeira a servirem de base e de paredes, sendo interligadas através de parafusos e esquadros de aço inoxidável. Nas faces superiores das paredes, foram aplicados um perfil de PVC de cor negra em cada uma das paredes, para melhor acomodar o *scanner*.

Podemos observar o resultado desta construção nas Figuras 4.1 e 4.2, ilustradas abaixo:

Figura 4.1 - Vista de cima da base construída para a caixa envolvente

Figura 4.2 - Vista lateral da base construída para a caixa envolvente

4.1.3 Resultado Final

Obtido o *scanner* e construída a caixa, procedeu-se então à montagem do protótipo da componente física do sistema proposto, como se pode observar abaixo na Figura 4.3:

Figura 4.3 - Vista de cima da montagem do protótipo do sistema físico

É de realçar também que na Figura acima, já se encontra a gaveta no seu interior, exemplificando uma amostra do seu funcionamento pretendido.

4.2 Softwares Adicionais

De modo a auxiliar o desenvolvimento do *software* principal e para obter melhores result ados no processamento das imagens digitalizadas e no reconhecimento da informação nelas contidas, foi necessário recorrer ao uso de alguns *softwares* adicionais, abaixo apresentados e a razão do seu uso explicada:

4.2.1 Tessnet2

O Tessnet2 é um motor de OCR *open-source*, baseado no Tesseract, para criar aplicações em ambientes .NET[7]. Foi concebido para facilitar a produção de mais aplicações OCR baseadas

[7] *Framework* da Microsoft, que inclui uma coleção gigante de bibliotecas de código, classes e funções, e que permite interoperabilidade entre várias linguagens de programação, possibilitando a sua execução simultânea dentro do Windows

em Tesseract, visto que, como abordado no capítulo 2.1.3.3, o Tesseract é um motor de OCR muito *bare-bones*, mas que privilegia a criação e partilha entre utilizadores.

No *site* oficial do Tessnet2 é incluído um exemplo básico de como começar a utilizar o Tessnet2 num projeto, em que é passada uma imagem pelo processo de OCR, e no final é retornado um conjunto de palavras e o valor de confiança com que elas foram reconhecidas – um valor auxiliar interessante para o utilizador melhor afinar o seu algoritmo. Abaixo podemos observar um exemplo disso mesmo, duma aplicação demo partilhada no *site* do Tessnet2, na Figura 4.4:

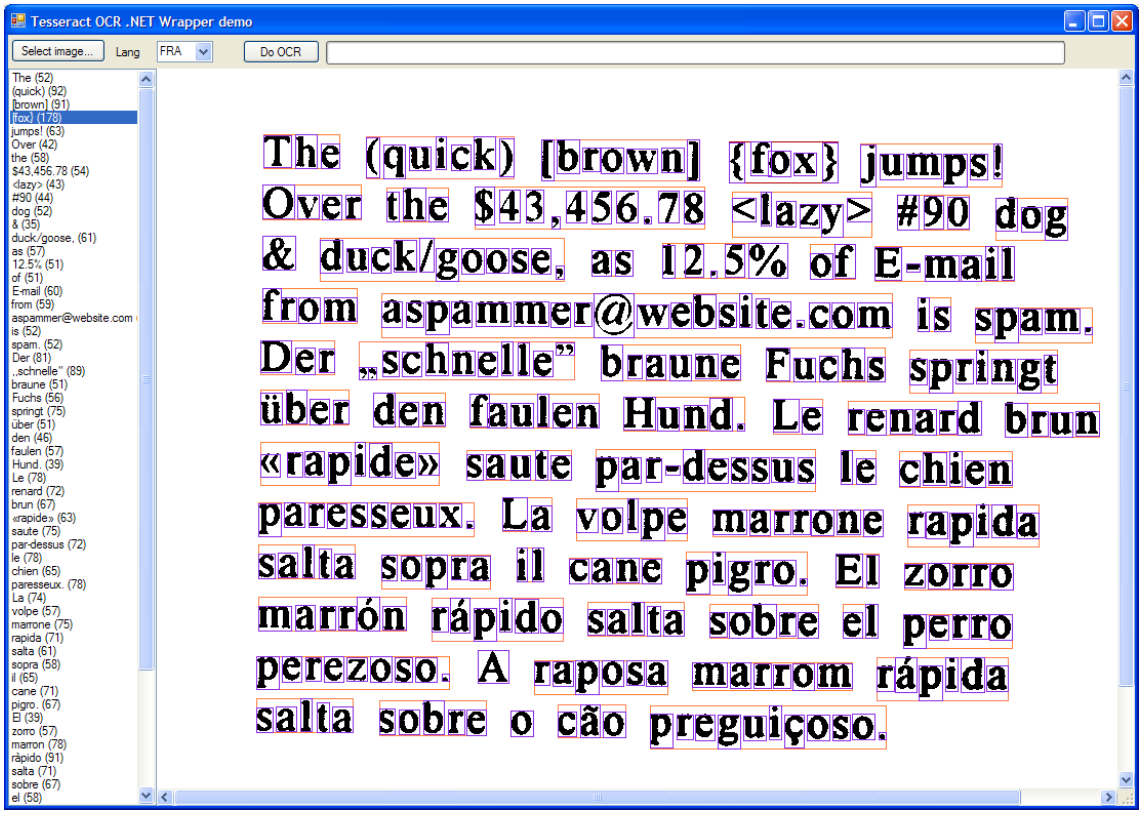


Figura 4.4 – *Print* de uma aplicação demo de OCR em Tessnet2 [34]

A demo consiste numa consola dividida em duas janelas: a janela da direita recebe uma qualquer imagem na qual será feito OCR – como se pode ver, todos os caracteres existentes estão delimitados por retângulos roxos, que denominam os caracteres individuais, e por retângulos laranjas, que denominam as palavras. Na janela da esquerda, estão registadas todas as palavras reconhecidas, cada uma com o seu devido valor de confiança.

Este valor de confiança pode tomar valores entre 0 e 255, sendo que o valor 0 significa que o reconhecimento foi excelente, enquanto que o valor 255 significa que o reconhecimento

foi horrível. Para o desenvolvimento deste projeto, o valor 150 foi tomado como um valor de *threshold* em que, qualquer valor de confiança acima deste seja sinalizado como tendo sido muito provavelmente mal reconhecido, e carece de atenção.

Uma vez que é fácil de entender e utilizar, e oferece ferramentas úteis para a resolução do problema, bem como o facto de poder ser utilizado em ambientes .NET, o Tessnet2 foi o motor de OCR escolhido para o desenvolvimento deste projeto.

4.2.2 EmguCV

O EmguCV é um *wrapper*[8] para ambientes .NET para a biblioteca de funções de processamento de imagem OpenCV. Estas funções são indispensáveis para o pré-processamento das imagens, de modo a torna-las mais apropriadas para o reconhecimento de caracteres, bem como para a gestão das variáveis de imagem dentro do *software* desenvolvido.

Na unidade curricular de Sistemas Sensoriais, o EmguCV é utilizado extensivamente devido ao conteúdo da cadeira, que incide fortemente no processamento de imagem. Tendo já concluído Sistemas Sensoriais, e sendo que o projeto final dessa unidade curricular acaba por ser uma espécie de precursor a este projeto, o EmguCV foi o escolhido para auxiliar no processamento de imagens neste projeto.

4.2.3 ZXing.NET

Um aspeto a considerar durante o desenvolvimento do projeto foi a existência de *QR codes* nos blocos histológicos: os códigos não servem um propósito meramente estético, mas sim informativo, visto que o código contém a mesma informação impressa na face do bloco, pelo que serve como fonte secundária de informação a reconhecer, caso o reconhecimento no bloco não seja sucedido.

No entanto, os motores de OCR apenas detetam caracteres, não estão equipados para detetar mais nada. Como tal, foi necessário averiguar algum *software* que permitisse o reconhecimento de códigos QR.

A escolha recaiu na biblioteca ZXing.NET, um *port* para ambientes .NET da biblioteca original ZXing, desenvolvida em Java, que permite a codificação e descodificação de uma larga gama de códigos de barras, nos quais se incluem os *QR codes*.

[8] Código que encapsula a funcionalidade de outro programa ou função, executando esse código utilizando o *wrapper* como uma espécie de interface, concedendo abstração ao código original e interoperabilidade ao sistema

Semelhantemente ao Tessnet2, é incluído no *site* oficial do ZXing.NET um exemplo básico de utilização do mesmo, em que é chamada uma função de descodificação que vai tentar descodificar uma imagem pré-existente, onde no final é apresentado ao utilizador o tipo de código e o conteúdo do mesmo, mediante o sucesso da descodificação.

4.3 *Software Principal*

Crucial ao sucesso do projeto, o desenvolvimento do *software* principal terá de conseguir fazer interface com o protótipo físico e permitir aos patologistas acesso a um processo simples e eficaz de digitalização da informação dos blocos e reconhecimento da mesma.

Intitulado de "Block Manager 2018", o programa foi criado no Visual Studio, um ambiente de desenvolvimento dedicado ao .NET e escrito na linguagem C#, de modo a acomodar a implementação dos *softwares* adicionais apresentados anteriormente, e o seu funcionamento é de centro de operações de todo o processo de reconhecimento, processamento e gestão dos blocos histológicos.

Na Figura 4.5 ilustrada abaixo, é possível observar o menu principal que surge quando o programa é inicializado:

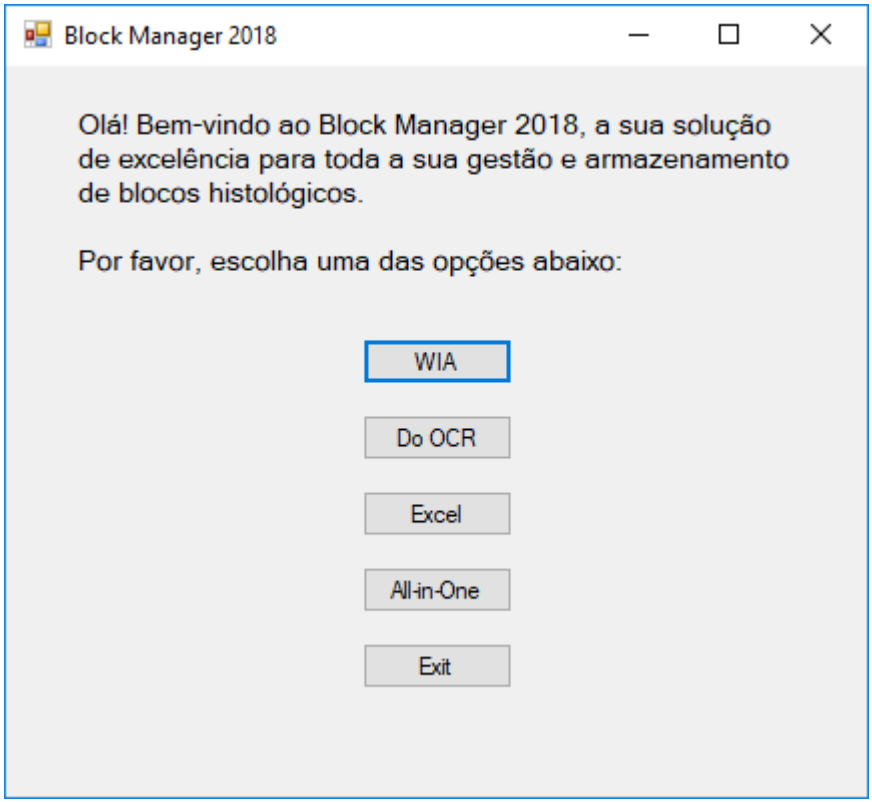


Figura 4.5 - Menu principal do *software* Block Manager 2018

Imediatamente, o utilizador pode observar 5 opções possíveis:

- Clicar no botão "WIA", que leva o utilizador a fazer a digitalização dos conteúdos inseridos no protótipo;
- Clicar no botão "Do OCR", que leva o utilizador a fazer OCR numa imagem à escolha, bem como guarda a informação reconhecida num ficheiro Excel;
- Clicar no botão "Excel", que abre uma janela que permite ao utilizador abrir, modificar e guardar qualquer ficheiro Excel que contenha as informações reconhecidas;
- Clicar no botão "All-in-One", que combina os botões "WIA" e "Do OCR", procedendo imediatamente ao processo de OCR após a digitalização efetuada;
- Clicar no botão "Exit" que, como o nome indica, sai do programa.

O programa foi concebido de modo a que fosse o mais automático possível, apenas precisando do *input* do utilizador nos casos mais dramáticos, em que o OCR tenha tido resultados péssimos, por exemplo. Se, no entanto, o OCR produzir resultados positivos, entre o primeiro

clique e os resultados serem guardados em Excel, o utilizador apenas terá que esperar uns segundos.

4.3.1 Classes

Na constituição do programa, este foi dividido em várias classes, em que cada uma opera uma parte distinta no funcionamento do programa. Estas classes serão abordadas a seguir:

4.3.1.1 RecognitionClass

Esta classe tem a tarefa de tratar de todos os processos relativos ao reconhecimento de caracteres, e é nesta que são efetuados todos os processos de OCR: extração da informação, tratamento e gravação da mesma.

Algumas funções importantes inerentes à RecognitionClass incluem as funções Do_OCR_Numbers/Do_OCR_Letters, que se encarregam de realizar o processo de OCR a uma qualquer secção da imagem, a função Perform_OCR, na qual são chamadas as funções Do_OCR várias vezes, para as várias secções do bloco histológico, ou a função Decode_QR, baseada no ZXing.NET, onde se procede à descodificação dos *QR codes*.

4.3.1.2 WIAClass

Esta classe faz a interface com o protocolo de aquisição de imagens do Windows, o WIA, e estabelece o contacto entre computador e *scanner*, ou seja, entre computador e protótipo, permitindo ao *software* aceder imediatamente à digitalização dos blocos.

Possui também uma função ShowErrorCode, que informa o utilizador de eventuais erros na ligação WIA entre computador e *scanner*, que, ao existir, inviabilizam a digitalização.

É de realçar ainda que o código implementado nesta classe foi baseado e adaptado dos tutoriais presentes em [35] e em [36].

4.3.1.3 ExcelClass

Esta classe contém todas as funções relativas à utilização do Excel dentro do *software*, permitindo a este realizar qualquer operação básica de Excel, como criar um ficheiro novo, escrever dentro dum ficheiro ou guardá-lo.

As funções existentes nesta classe são utilizadas dentro do funcionamento principal da RecognitionClass, em que o resultado do processo de OCR é finalmente guardado para um ficheiro Excel.

É de realçar ainda que o código implementado nesta classe foi baseado e adaptado dos tutoriais presentes em [37][38][39][40].

4.3.1.4 ExcelViewer

Esta classe permite ao utilizador o acesso às funcionalidades do Excel dentro do Block Manager, funcionando como uma ligação entre os dois programas.

É através desta classe que o Block Manager pode servir como centro de requisição e gestão, visto que os utilizadores poderão facilmente aceder aos ficheiros e saber a localização dos blocos, bem como marcá-los como requisitados, podendo até deixar informações como quem requisitou o bloco e até quando.

É de realçar ainda que o código implementado nesta classe foi baseado e adaptado do tutorial presente em [41].

4.3.1.5 DrawerNameForm e DrawerNumForm

Estas duas classes têm uma funcionalidade e objetivo igualmente simples: uma vai pedir ao utilizador o número de gavetas que vão ser inseridas, a outra pede o nome de cada gaveta que será inserida. Este controlo é feito com a presença do utilizador de modo a minimizar os erros do processo a reconhecer as gavetas nas imagens.

Podemos observar abaixo nas Figuras 4.6 e 4.7 os *forms* das classes:

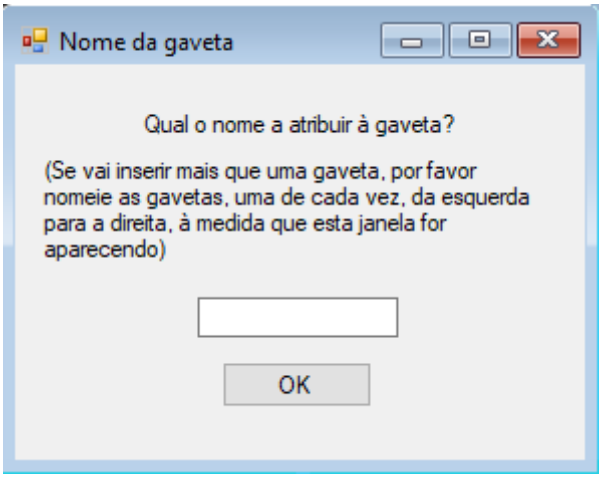


Figura 4.6 - Ilustração de DrawerNameForm

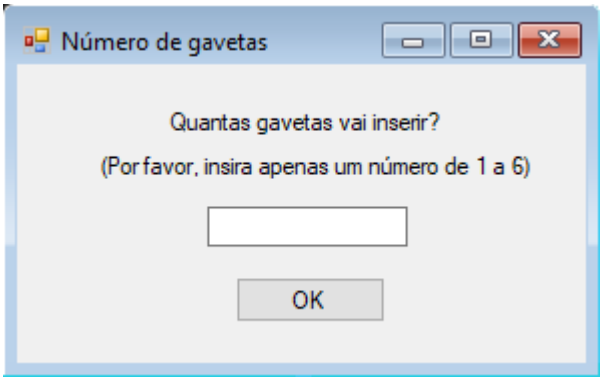


Figura 4.7 - Ilustração de DrawerNumForm

4.3.1.6 ImageClass

Esta classe contém todas as funções relativas ao processamento de imagem dentro do *software*, sendo que três são do foro geral do processamento de imagem – Mean_NxN, ConvertToBW e ConvertToBW_Otsu – e as restantes foram criadas especificamente para o processamento das imagens retiradas do *scanner* inserido no protótipo, como por exemplo: a função locateLimits, que permite ao utilizador saber onde se localizam os limites laterais de uma qualquer gaveta, ou a função locateBlocks, que guarda as coordenadas de cada bloco histológico.

As funções existentes nesta classe são utilizadas dentro do funcionamento principal da RecognitionClass, mais especificamente no pré-processamento da imagem.

4.3.1.7 ImageViewer

Esta classe permite ao utilizador visionar qualquer variável de imagem presente no programa a qualquer altura. O seu uso incide mais na fase de testes do programa do que na fase de uso.

4.3.1.8 Proofreader

Esta classe permite ao utilizador efetuar correções nos resultados obtidos pelo OCR: se, a qualquer altura, algum resultado tiver um valor de confiança acima de 150, o programa sinaliza esse resultado, e é nesta classe que são tratados esses possíveis erros.

A Figura 4.8 abaixo exemplifica a estrutura do *form* da classe:

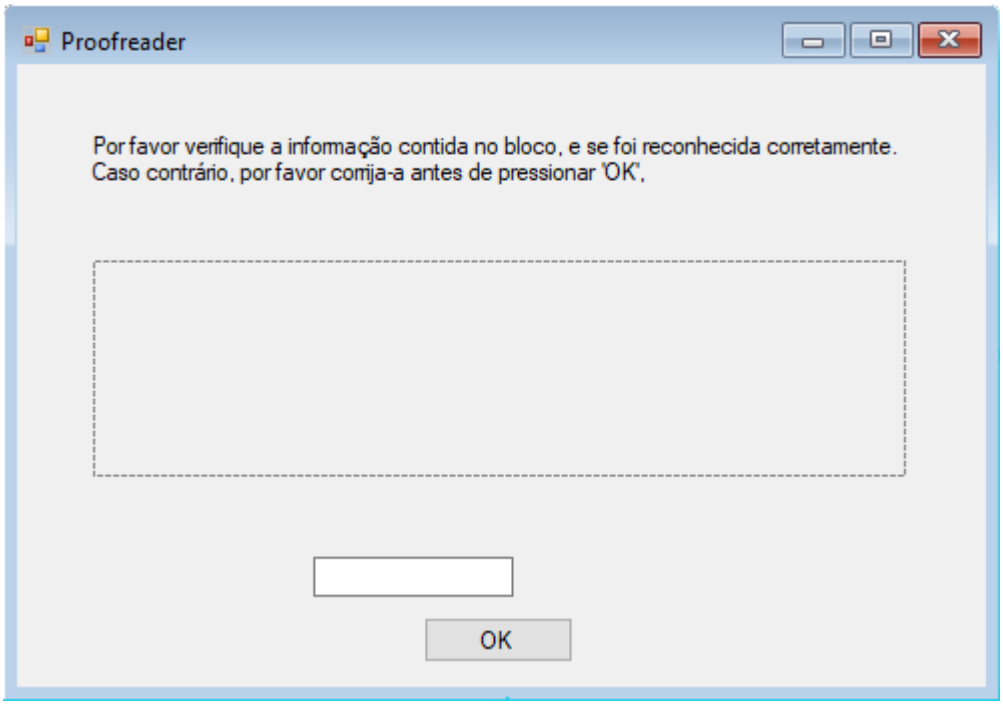


Figura 4.8 - Ilustração de Proofreader

Durante o normal funcionamento do programa, o bloco em questão com a informação sinalizada será impresso dentro do tracejado central da janela, bem como a palavra reconhecida dentro da caixa de texto, sendo que à sua frente o utilizador será relembrado qual o campo do bloco de que se trata. Caberá então ao utilizador verificar se a informação foi bem reconhecida ou não: se foi, basta carregar no botão "OK" para seguir em frente, caso contrário, deverá corri-

gir o conteúdo da caixa de texto antes de carregar em "OK", o que irá atualizar a informação reconhecida.

4.3.2 Funcionamento Geral

Nesta secção, será exposto o funcionamento geral que o programa deverá levar, de início ao fim, no reconhecimento de uma única gaveta dentro do protótipo.

Executando o programa, o utilizador é gracejado com a janela do menu principal, como ilustrada na Figura 4.5. O funcionamento geral do programa, partindo deste ponto de início, pode ser resumido pelo fluxograma retratado abaixo na Figura 4.9:

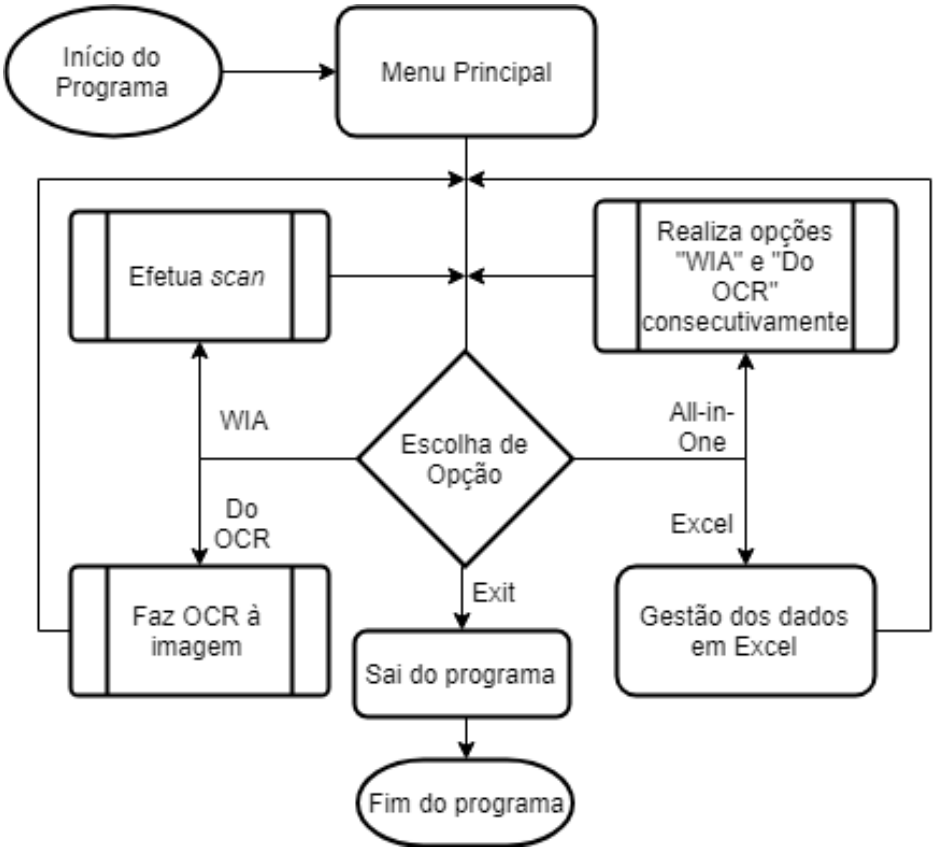


Figura 4.9 – Fluxograma referente ao funcionamento geral do Block Manager 2018

Entrando em mais detalhe, o primeiro passo será a realização de uma digitalização, de modo a obter uma imagem para tentar reconhecer. Assim sendo, o processo de digitalização, representado como opção na Figura 4.9 acima, é exibido no fluxograma abaixo, referente à Figura 4.10:

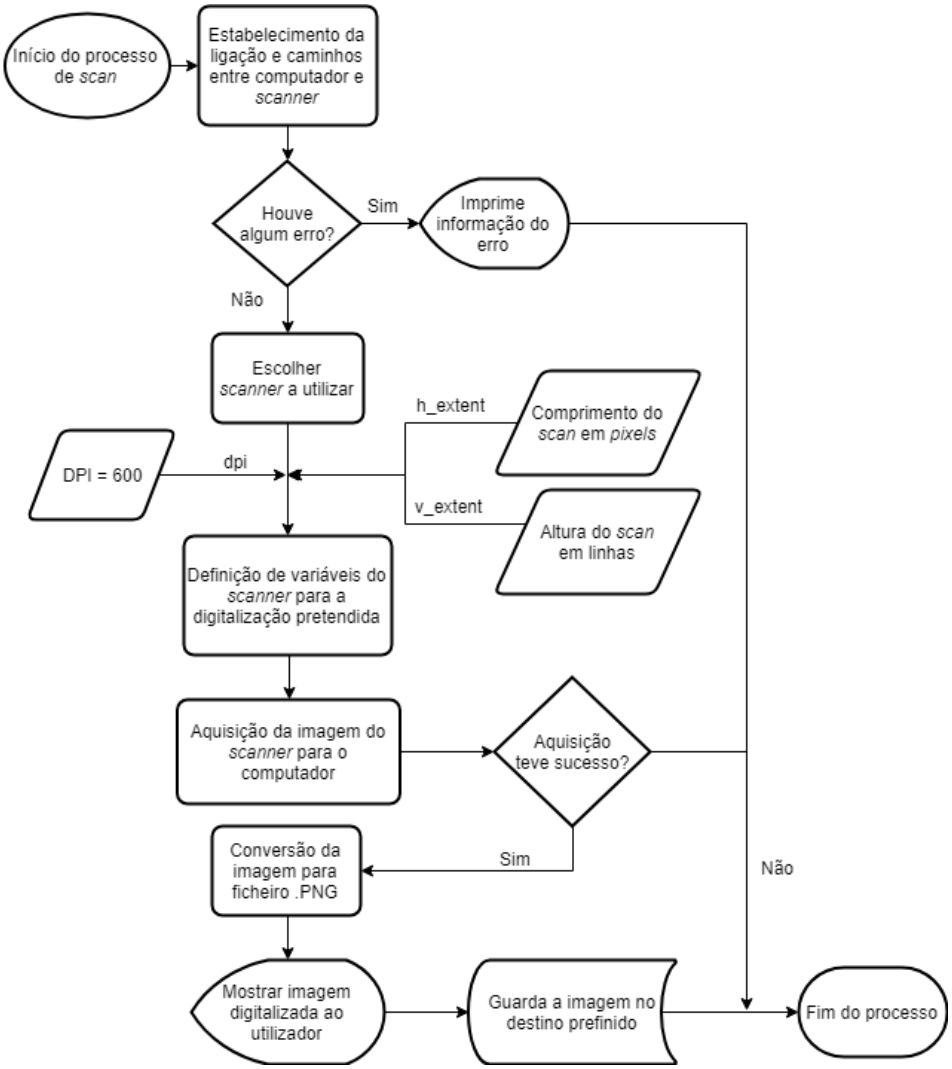


Figura 4.10 – Fluxograma referente ao funcionamento do processo de digitalização

Em primeiro lugar, seguindo o fluxograma, é estabelecida a ligação computador-*scanner*, em que o programa estabelece alguns valores de definições por cima das pré-existentes do *scanner,* como mudar o valor de DPI da foto a ser digitalizada, ou o esquema de cores em que esta é tirada. Neste caso, para facilitar o pré-processamento mais tarde, a foto é forçada a ser digitalizada a tons de cinzento.

Após a ligação estar estabelecida, não havendo nenhum erro, o *scanner* procede à digitalização do conteúdo inserido no protótipo, e o programa mostra ao utilizador o que foi digitalizado, através da classe ImageViewer, antes de guardar a imagem como ficheiro.

É possível observar abaixo na Figura 4.11 um exemplo de digitalização:

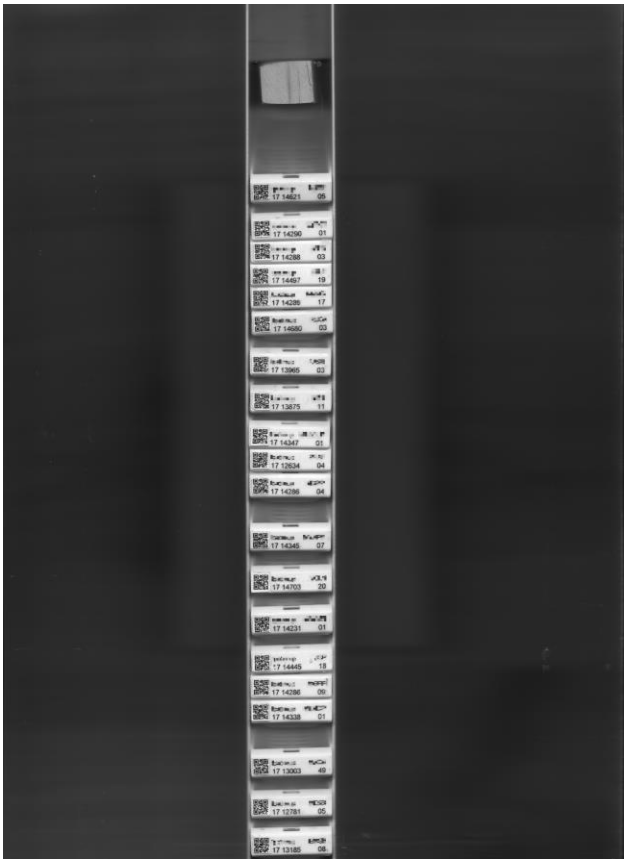


Figura 4.11 - Exemplo de interior do compartimento de digitalização

De realçar que a figura acima representada foi adaptada de modo a manter o anonimato do instituto e dos seus pacientes. Obtendo uma imagem de qualidade aceitável, procede-se ao reconhecimento da mesma. Este processo é representado de um modo sucinto na Figura 4.12, e descrito em mais detalhe de seguida.

A imagem é pré-processada, sendo-lhe passada um filtro de média 3x3 para limpar o ruído da imagem e é posteriormente convertida para uma imagem binária através do método de Otsu. É então chamada a função locateLimits, que irá encontrar as coordenadas dos limites laterais da gaveta, e dividindo-a em duas áreas: qr_area, referente à área apenas dos *QR codes* e blocks_area, referente à restante área de bloco.

De seguida, é chamada a função locateBlocks, que irá encontrar a localização de cada bloco dentro da gaveta, que serão partilhadas mais à frente por vários *arrays* referentes às subsecções dos blocos. Entretanto, é chamada Decode_QR para reconhecer a informação contida nos *QR codes*, guardando-a como auxiliar ao reconhecimento principal.

É chamada a função Fill_Blocks, que preenche os vários *arrays* das subsecções dos blocos com as suas coordenadas correspondentes, seguida da função Perform_OCR, na qual, como o nome indica, se realiza o processo de OCR para cada bloco. O *array* blocks_info irá receber toda esta informação, que terá de ser revista e, eventualmente, corrigida.

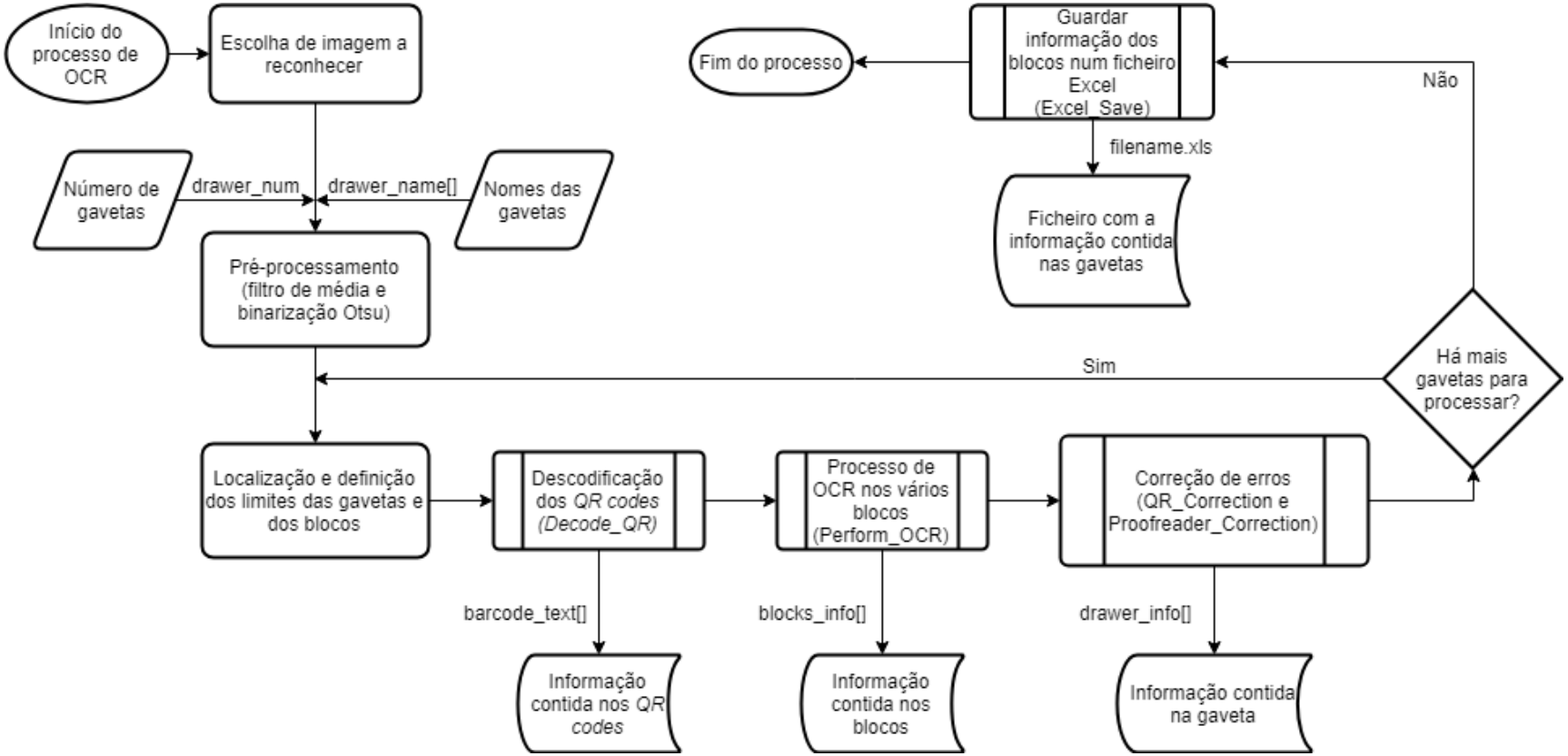


Figura 4.12 – Fluxograma referente ao funcionamento do processo de OCR

Como tal, são chamadas as duas funções de correção: QR_Correction e Proofreader_Correction, sendo que na primeira, o programa irá automaticamente verificar, através da informação contida nos *QR codes*, potenciais erros nas informações dos blocos. Só depois, através da segunda função, o utilizador poderá ser abordado para interagir na correção. Podemos ver um exemplo disso mesmo na Figura 4.13 abaixo:

Figura 4.13 - Exemplo de Proofreader no funcionamento geral

De realçar que a imagem é adaptada, devido à existência do nome real do instituto em questão nos blocos histológicos, que aqui é denominado por IVT, bem como informação que possa identificar um paciente do IVT.

Feitas as correções, a informação é então guardada num ficheiro Excel, facilmente acessível mesmo dentro do programa, que toca um sinal sonoro, alertando o utilizador do final do processo, estando pronto para reconhecer mais gavetas.

Para leitores mais confusos, ou mais interessados, estão disponíveis no final deste documento, na secção dos Anexos, mais alguns fluxogramas representativos do funcionamento de outros aspetos do programa, bem como o código implementado de algumas classes e/ou funções mais relevantes para a execução do mesmo.

5

Resultados e Discussão

Neste capítulo, serão apresentados os resultados provenientes dos testes efetuados ao programa desenvolvido. Os testes incidem na validação das capacidades propostas do sistema, nomeadamente a deteção das gavetas, a deteção dos blocos, a deteção dos *QR codes*, o reconhecimento através de OCR da informação contida nos blocos e nos *QR codes* e o registo dessa informação num ficheiro à parte. Estes são os pontos fulcrais para a validação da hipótese, havendo depois o estudo das consequências de variações nas características das gavetas, que poderão mudar os resultados obtidos.

5.1 Teste de uma única gaveta em condições favoráveis

Este teste serve como uma espécie de teste de "controlo", servindo também como base de comparação para os vários testes realizados.

Primeiramente, é testada uma única gaveta, situada sensivelmente a meio do compartimento, com 20 blocos histológicos no seu interior, tendo estes sido colocados com um maior cuidado na sua disposição, de modo a tentar obter os melhores resultados.

Insere-se a sua imagem, previamente retirada, na funcionalidade de OCR do programa e, com o auxílio da classe ImageViewer, bem como através das funcionalidades do Visual Studio, torna-se possível ir acompanhando o progresso do programa.

O primeiro ponto a validar é a capacidade ou não de detetar a gaveta dentro do compartimento. Assumindo que a imagem inserida é semelhante à Figura 4.11, teremos um resultado também semelhante à Figura 5.1, ilustrada abaixo:

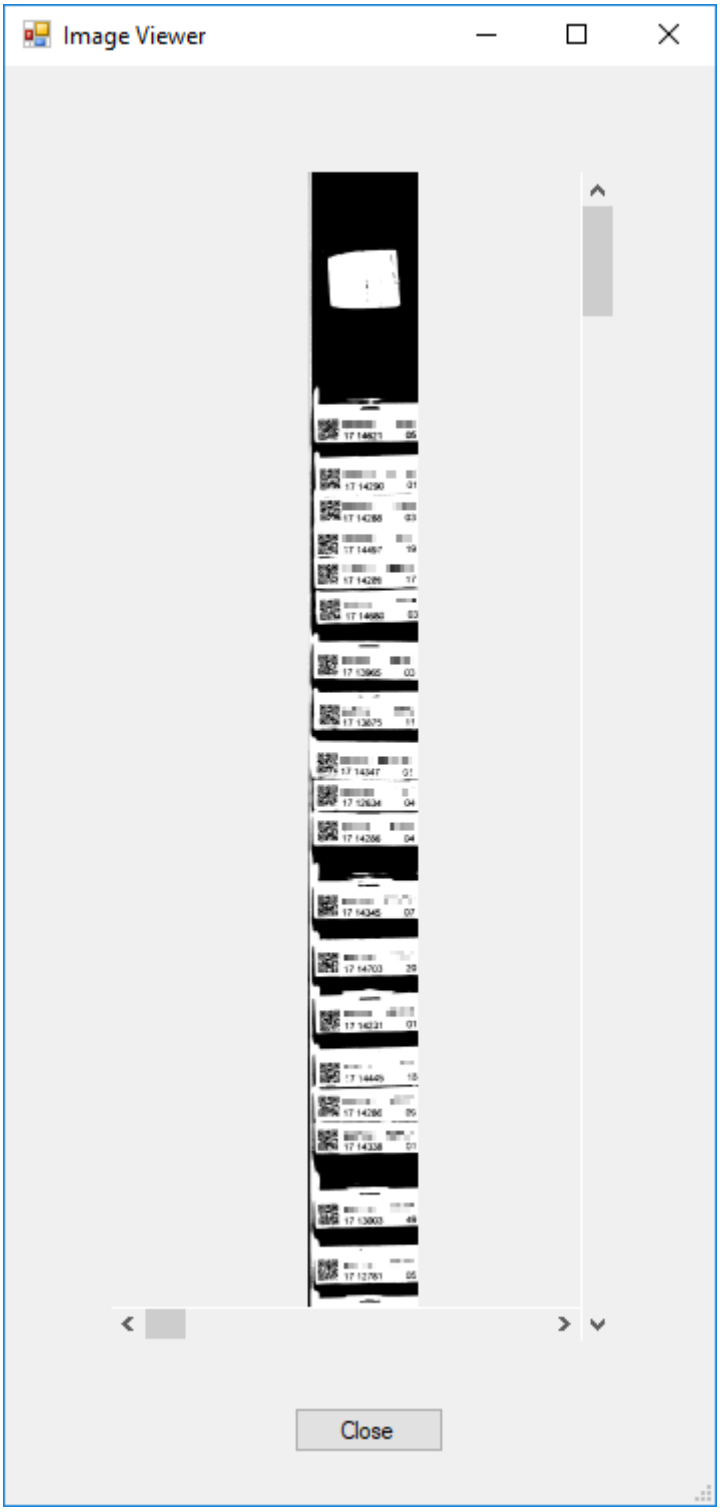


Figura 5.1 - Visualização em Visual Studio da área da gaveta, detetada automaticamente

De realçar que a imagem acima é adaptada, devido à existência do nome real do instituto em questão nos blocos histológicos, bem como informação que possa identificar um paciente do instituto.

Como se pode observar, a deteção foi realizada com sucesso, descartando o espaço vazio presente dentro do compartimento, imprimindo para o ecrã apenas a zona da gaveta. De seguida, é necessário separar as duas zonas de *QR codes* e de bloco restante. Partindo da Figura 5.1, obtemos o resultado demonstrado abaixo na Figura 5.2 (adaptada, como descrito anteriormente) :

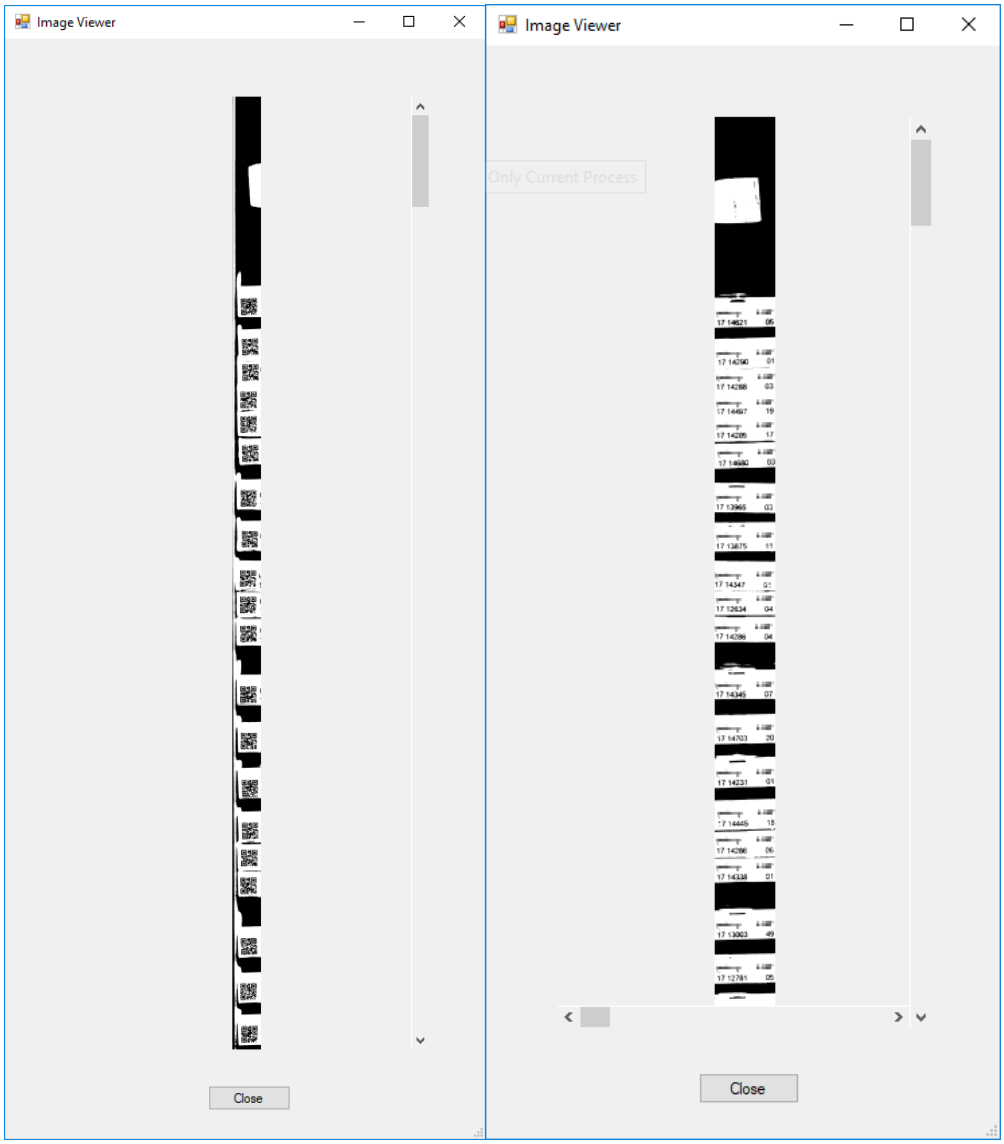


Figura 5.2 - Visualização em Visual Studio da a) área dos *QR codes* e b) da área do corpo principal dos blocos, ambas detetadas automaticamente

Uma vez detetadas ambas as áreas, proceder-se-á à localização dos blocos e dos *QR codes* isolados. Podemos observar nas Figuras 5.3 e 5.4 o resultado da localização do corpo de um bloco e de um *QR code*, respetivamente, servindo como exemplo para os restantes:

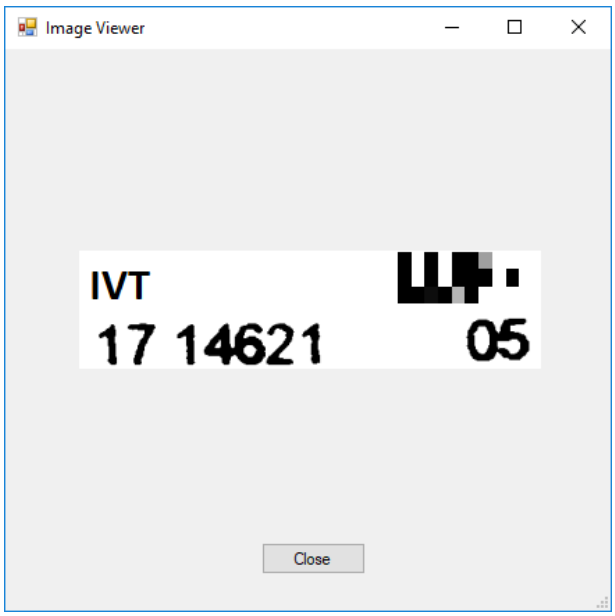
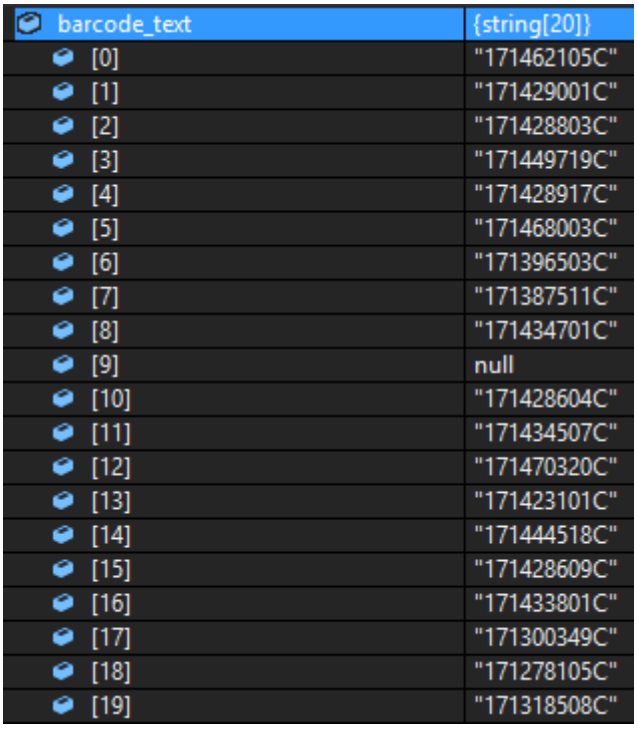


Figura 5.3 - Visualização em Visual Studio do corpo principal de um bloco, detetado automaticamente

Figura 5.4 - Visualização em Visual Studio de um QR code, detetado automaticamente

De realçar, mais uma vez, que na Figura 5.3, houve uma adaptação de modo à mudança do nome real do instituto, que para os efeitos desta tese, é denominado de IVT, bem como para ocultar qualquer informação relacionada à identidade de um paciente.

O programa irá, de seguida, proceder à realização do OCR em cada uma das coordenadas dos *QR codes* e dos blocos. Para o reconhecimento dos *QR codes*, é possível verificar o resultado através das funcionalidades do Visual Studio, a meio do processo. A Figura 5.5 abaixo demonstra o resultado obtido:



barcode_text	{string[20]}
[0]	"171462105C"
[1]	"171429001C"
[2]	"171428803C"
[3]	"171449719C"
[4]	"171428917C"
[5]	"171468003C"
[6]	"171396503C"
[7]	"171387511C"
[8]	"171434701C"
[9]	null
[10]	"171428604C"
[11]	"171434507C"
[12]	"171470320C"
[13]	"171423101C"
[14]	"171444518C"
[15]	"171428609C"
[16]	"171433801C"
[17]	"171300349C"
[18]	"171278105C"
[19]	"171318508C"

Figura 5.5 - Resultados obtidos do OCR realizado nos QR codes

A informação obtida dos *QR codes* vem no formato "XXYYYYYZZC", em que XX é referente ao número encontrado na região inferior esquerda do bloco, YYYYY é referente ao número encontrado na região inferior média do bloco, ZZ é referente ao número encontrado na região inferior direita do bloco e C é um caracter existente na criação do *QR code*.

Como podemos observar, o reconhecimento dos *QR codes* é bem-sucedido, reconhecendo com sucesso 19 em 20 códigos.

Completado o processo de OCR nos blocos, estes poderão ter ainda erros nos resultados obtidos, que podem derivar de uma localização mal determinada ou desgaste dos caracteres impressos nos blocos. Como tal, é geralmente feito um processo de correção através dos resultados obtidos na Figura 5.5 e através da funcionalidade do Proofreader, ilustrada na Figura 5.6 a seguir. No entanto, para efeitos de estudo dos resultados, não se procederá à correção do Proofreader, de modo a perceber a qualidade do algoritmo sem interferência humana. Mais uma vez, visto que contém a representação de um bloco histológico, a imagem foi adaptada para mostrar IVT, ao invés do nome real do instituto, bem como para ocultar a identidade do paciente.

Figura 5.6 - Visualização do Proofreader no Visual Studio

Finalmente, após as correções serem feitas (ou não), o programa irá registar as informações contidas nos blocos num ficheiro Excel, para melhor visualização e gestão por parte do utilizador. Os resultados, por corrigir, podem ser observados abaixo na Figura 5.7:

Instituto	Nome	Ano	Número	Tipo	Requisitado?
IVT	****	17	14621	05	Não
IVT Errado	****	17	14290	01	Não
IVT	****	17	14288	03	Não
IVT Errado	****	17	14497	19	Não
IVT Errado	****	17	14289	17	Não
IVT	****	17	14680	03	Não
IVT Errado	****	17	13965	03	Não
IVT	****	17	13875	11	Não
IVT Errado	****	17	14347	01	Não
IVT	****	17	12634	04	Não
IVT	****	17	14286	04	Não
IVT	****	17	14345	07	Não
IVT	****	17	14703	20	Não
IVT	****	17	14231	01	Não
IVT Errado	****	17	14445	18	Não
IVT Errado	****	17	14286	09	Não
IVT	****	17	14338	01	Não
IVT Errado	****	17	13003	49	Não
IVT Errado	****	17	12781	05	Não
IVT Errado	****	17	13185	08	Não

Figura 5.7 - Resultados obtidos através do processo de OCR oriundos dos blocos, visualizados em Excel

Como podemos observar na imagem, na coluna "Instituto", os resultados obtidos relativamente ao instituto do qual o bloco provém foram, para efeitos desta tese, alterados para "IVT" onde o reconhecimento tenha sido correto, e para "IVT Errado" onde o reconhecimento não tenha sido correto. Na coluna "Nome", os resultados obtidos foram também estes alterados para "****", de modo a proteger a identidade dos pacientes. Estas adaptações serão recorrentes no corpo do capítulo 5.

Comparando os resultados obtidos com os resultados esperados, podemos redigir a seguinte tabela de eficácia:

Tabela 5.1 - Eficácia de resultados do teste com uma gaveta

Instituto	Nome	Ano	Número	Tipo	Total
10/20	18/20	20/20	20/20	20/20	88/100
50%	90%	100%	100%	100%	88%

Observando os resultados, é possível tirar as seguintes ilações: o programa é capaz de detetar a gaveta, os blocos e os *QR codes*, conseguindo posteriormente reconhecer e registar a informação digitalizada num ficheiro à parte, concretizando os objetivos base.

No geral, o programa realiza um processo que leva a um reconhecimento bastante bom, tendo atingido uma taxa de acerto de 88% em 100 itens a reconhecer e categorizar. No entanto, essa taxa de acerto é decrescida pela taxa de acerto particular do campo do "Instituto", que reconheceu apenas 50% das palavras com sucesso. Isto deve-se, maioritariamente, ao desgaste encontrado na impressão do nome do instituto nos blocos, o que levou ao OCR a não conseguir reconhecer totalmente o nome, embora, no geral, conseguisse reconhecer grande parte da palavra.

Em relação à taxa de acerto do campo do "Nome", esta não reconhece apenas 2 nomes, que, neste caso, se devem à existência de algum ruído que não foi totalmente limpo nem segmentado, o que significa que, embora o algoritmo apresente bons resultados, ainda está longe de ser considerado perfeito.

5.2 *Teste de uma única gaveta em localizações diferentes*

Este teste irá comparar os resultados obtidos em 5.1 com os resultados obtidos através do teste de uma única gaveta, cuja captação digital foi realizada com a gaveta a encontrar-se em localizações diferentes: no espaço mais à esquerda, mais à direita e no centro da caixa envolvente.

O objetivo é averiguar se o programa produz resultados diferentes, numa mesma gaveta, com os blocos dispostos da mesma forma, para disposições diferentes dentro da caixa envolvente.

Para efeitos de síntese, e de modo a evitar redundância, os resultados da gaveta no centro da caixa são tidos como equivalentes aos obtidos em 5.1.

Tal como em 5.1, o processo que se realiza é o mesmo, pelo que se irá apenas demonstrar a gaveta bem localizada e os resultados obtidos, que podem ser observados abaixo nas Figuras 5.8 - 5.10, respetivamente, que foram adaptadas como descrito anteriormente:

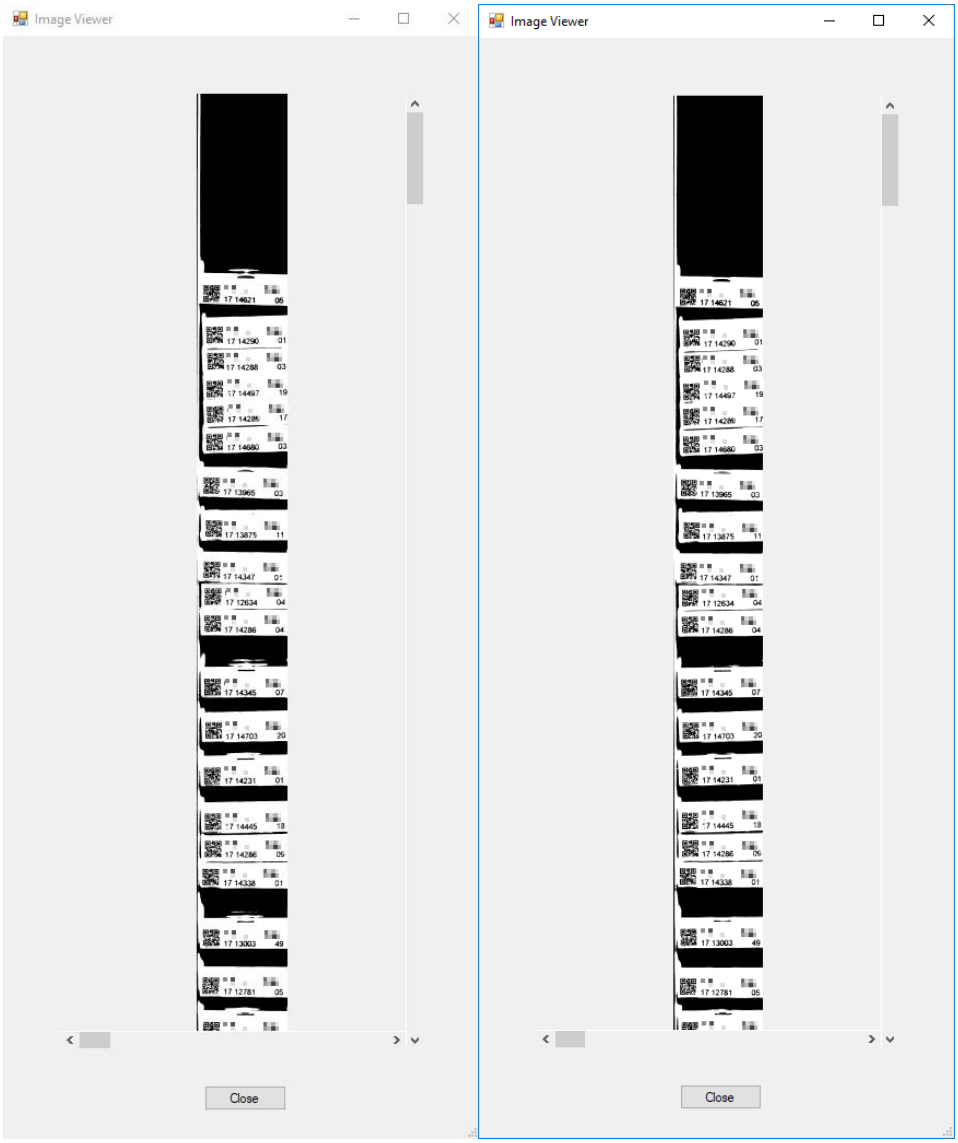


Figura 5.8 - Visualização em Visual Studio da área da gaveta, detetada automaticamente, a) à esquerda e b) à direita

Instituto	Nome	Ano	Número	Tipo	Requisitado?
IVT	****	17	14621	05	Não
IVT Errado	****	17	14290	01	Não
IVT Errado	****	17	14288	03	Não
IVT Errado	****	17	14497	19	Não
IVT Errado	****	17	14289	17	Não
IVT	****	17	14680	03	Não
IVT Errado	****	17	13965	03	Não
IVT	****	17	13875	11	Não
IVT Errado	****	17	14347	01	Não
IVT Errado	****	17	12634	04	Não
IVT Errado	****	17	14286	04	Não
IVT	****	17	14345	07	Não
IVT	****	17	14703	20	Não
IVT	****	17	14231	01	Não
IVT Errado	****	17	14445	18	Não
IVT Errado	****	17	14286	09	Não
IVT Errado	****	17	14338	01	Não
IVT Errado	****	17	13003	49	Não
IVT	****	17	12781	05	Não
IVT Errado	****	17	13185	08	Não

Figura 5.9 - Resultados obtidos através do processo de OCR oriundos dos blocos, visualizados em Excel, da gaveta à esquerda

Instituto	Nome	Ano	Número	Tipo	Requisitado?
IVT	****	17	14621	05	Não
IVT Errado	****	17	14290	01	Não
IVT Errado	****	51	714281	03	Não
IVT Errado	****	17	14497	19	Não
IVT Errado	****	17	14289	17	Não
IVT Errado	****	17	14680	03	Não
IVT	****	17	13965	03	Não
IVT	****	17	13875	11	Não
IVT Errado	****	17	14347	02	Não
IVT	****	17	12634	04	Não
IVT Errado	****	17	14286	04	Não
IVT	****	17	14345	07	Não
IVT Errado	****	17	14703	20	Não
IVT	****	17	14231	01	Não
IVT Errado	****	17	14445	18	Não
IVT Errado	****	17	14286	09	Não
IVT	****	17	14338	01	Não
IVT Errado	****	17	13003	49	Não
IVT Errado	****	17	12781	05	Não
IVT Errado	****	17	13185	08	Não

Figura 5.10 - Resultados obtidos através do processo de OCR oriundos dos blocos, visualizados em Excel, da gaveta à direita

Comparando os resultados obtidos com os esperados, podemos uma vez mais redigir as seguintes tabelas de eficácia:

Tabela 5.2 - Eficácia de resultados do teste com uma gaveta à esquerda

Instituto	Nome	Ano	Número	Tipo	Total
7/20	19/20	20/20	20/20	20/20	86/100
35%	95%	100%	100%	100%	86%

Tabela 5.3 - Eficácia de resultados do teste com uma gaveta à direita

Instituto	Nome	Ano	Número	Tipo	Total
7/20	19/20	19/20	19/20	19/20	83/100
35%	95%	95%	95%	95%	83%

Observando os resultados, podemos inferir de que a localização da gaveta dentro da caixa envolvente não causa grandes variações nos resultados obtidos, sendo todos eles muito próximos: 88% para a gaveta no meio, 86% para a gaveta à esquerda e 83% para a gaveta à direita, sendo que a maior parte dos erros são semelhantes: a erosão no campo do "Instituto" tornou muito difícil o seu reconhecimento automático, sendo necessário uma posterior intervenção através do Proofreader.

De resto, no mínimo todos os campos reconheceram 19 itens em 20, pelo que se pode considerar o reconhecimento como tendo sido bem sucedido. Ainda assim, deve-se destacar um erro na digitalização na gaveta à direita, no bloco com o nome "****", em que o seu ano foi reconhecido como "51" em vez de "17", tendo o seu número sido reconhecido como "714281" em vez de "14288".

Esta separação do caracter "7" em "17" e posterior aglutinação para o campo dos números dever-se-á a uma falha na localização correta dos limites dos campos do bloco. Já os caracteres "5" e "1" terão aparecido como fruto de ruído ou de uma segmentação menos eficaz.

5.3 Teste de várias gavetas ao mesmo tempo

Este teste irá comparar os resultados obtidos em 5.1 com os resultados obtidos através do teste de várias gavetas em simultâneo.

O objetivo é averiguar se o programa produz resultados diferentes, dependendo do número de gavetas que estão inseridas dentro da caixa envolvente.

5.3.1 Teste de duas gavetas

Neste subcapítulo, é abordado o teste efetuado a uma imagem do interior do compartimento do protótipo, contendo duas gavetas com blocos no seu interior.

Este teste não é, no entanto, totalmente fidedigno, visto que a imagem a analisar é uma montagem criada pela junção de duas imagens de uma gaveta sozinha. Foi necessário proceder deste modo para testar o reconhecimento de mais que uma gaveta, visto que o IVT apenas pôde emprestar uma gaveta com 20 blocos, pelo que foi necessário proceder deste modo para poder fazer pelo menos uma "simulação" do que seria se houvessem mais gavetas para testar.

Posto isto, o funcionamento geral é em tudo semelhante ao teste com uma gaveta, simplesmente repete-se mais uma vez após o primeiro teste, sendo que a única diferença reside na localização das gavetas: como agora existe mais que uma, é necessário encontrar as restantes, cuja localização depende da anterior. O resultado da localização das duas gavetas pode ser conferido abaixo, na Figura 5.11 que, como tem sido recorrente neste capítulo, foi adaptada de modo a ocultar informações sensíveis:

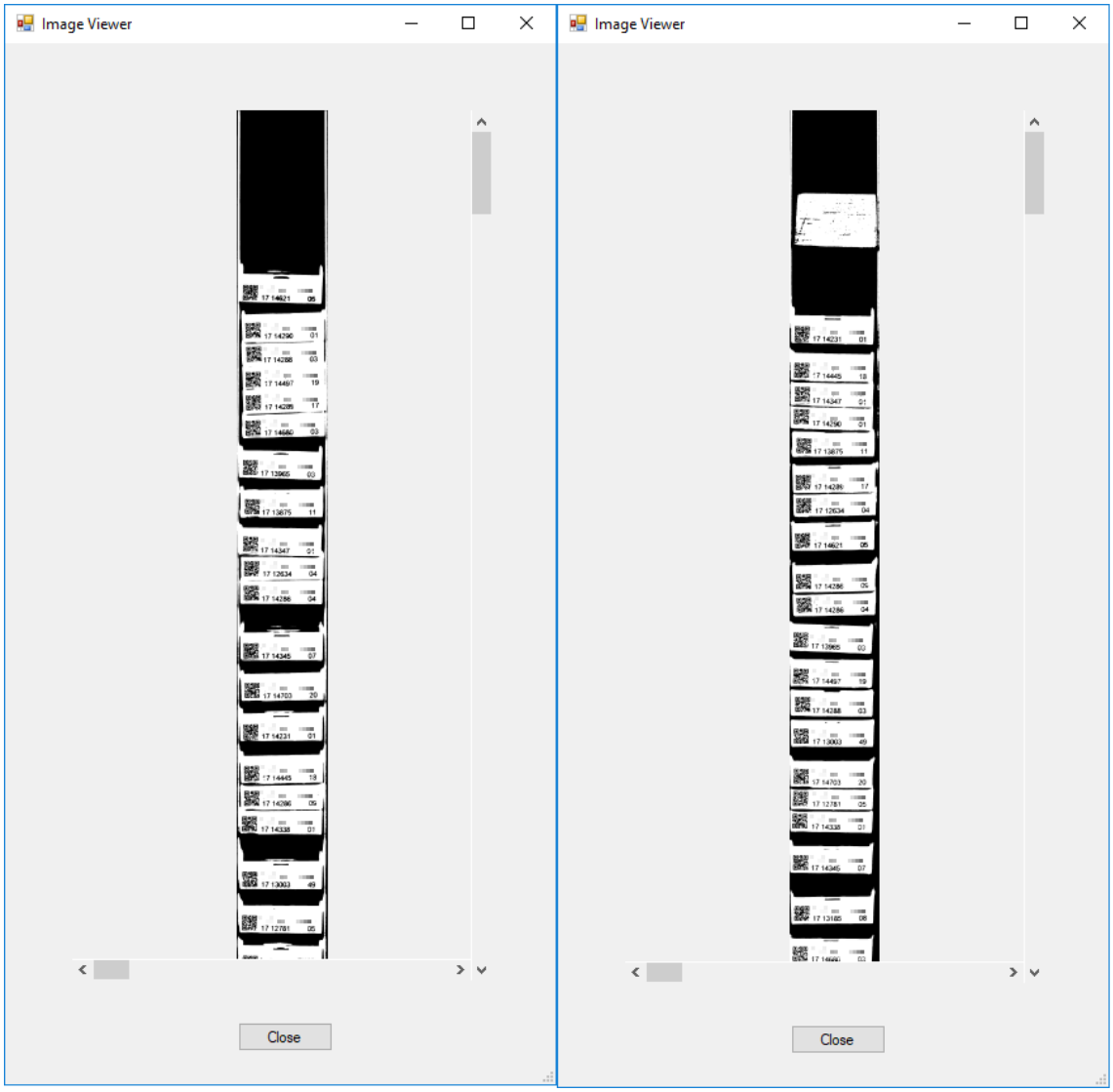


Figura 5.11 - Visualização das áreas das duas gavetas

O procedimento até ao registo da informação é análogo ao efetuado no capítulo 5.1, pelo que podemos saltar esse processo, já demonstrado acima, passando a demonstrar abaixo aos resultados obtidos e as tabelas de eficácia correspondentes:

Instituto	Nome	Ano	Número	Tipo	Requisitado?
IVT	****	17	14621	05	Não
IVT Errado	****	17	14290	01	Não
IVT Errado	****	17	14288	03	Não
IVT Errado	****	17	14497	19	Não
IVT Errado	****	17	14289	17	Não
IVT Errado	****	17	14680	03	Não
IVT Errado	****	17	13965	03	Não
IVT	****	17	13875	11	Não
IVT Errado	****	17	14347	01	Não
IVT Errado	****	17	12634	04	Não
IVT	****	17	14286	04	Não
IVT	****	17	14345	07	Não
IVT	****	17	14703	20	Não
IVT	****	17	14231	01	Não
IVT Errado	****	17	14445	18	Não
IVT Errado	****	17	14286	09	Não
IVT Errado	****	17	14338	01	Não
IVT Errado	****	17	13003	49	Não
IVT	****	17	12781	05	Não
IVT Errado	****	17	13185	08	Não

Figura 5.12 - Resultados obtidos através do processo de OCR oriundos dos blocos, visualizados em Excel, da 1ª gaveta

Instituto	Nome	Ano	Número	Tipo	Requisitado?
IVT Errado	****	17	14231	01	Não
IVT Errado	****	17	14445	18	Não
IVT Errado	****	7	14347	07	Não
IVT Errado	****	71	4290	01	Não
IVT Errado	****	17	13875	11	Não
IVT Errado	****	17	14289	17	Não
IVT Errado	****	17	12634	04	Não
IVT Errado	****	17	14621	05	Não
IVT Errado	****	17	14286	09	Não
IVT Errado	****	17	14286	04	Não
IVT Errado	****	71	3965	03	Não
IVT Errado	****	17	14497	19	Não
IVT Errado	****	71	4288	03	Não
IVT Errado	****	7	13003	49	Não
IVT Errado	****	71	14703	20	Não
IVT Errado	****	17	12781	05	Não
IVT Errado	****	71	4338	01	Não
IVT Errado	****	71	4345	07	Não
IVT Errado	****	71	13185	08	Não
IVT Errado	****	71	4680	02	Não

Figura 5.13 - Resultados obtidos através do processo de OCR oriundos dos blocos, visualizados em Excel, da 2ª gaveta

Tabela 5.4 - Eficácia de resultados da primeira gaveta do teste com duas gavetas

Instituto	Nome	Ano	Número	Tipo	Total
7/20	18/20	20/20	20/20	20/20	85/100
35%	90%	100%	100%	100%	85%

Tabela 5.5 - Eficácia de resultados da segunda gaveta do teste com duas gavetas

Instituto	Nome	Ano	Número	Tipo	Total
0/20	16/20	10/20	14/20	18/20	58/100
0%	80%	50%	70%	90%	58%

Os resultados da primeira gaveta são muito semelhantes aos obtidos na Tabela 5.1, o que seria de esperar, até que porque foi utilizada a mesma imagem nas duas. O que imediatamente saltará à vista são os resultados obtidos na segunda gaveta, uma taxa de acerto muito modesta de 58%.

Algumas razões podem ser apontadas para esta discrepância de uma gaveta para outra: o algoritmo de localização de gavetas pode não estar totalmente otimizado para a localização de mais de uma gaveta, visto que, ao encontrar as gavetas subsequentes a partir das anteriores, introduz erro a cada gaveta que localiza, podendo comprometer a boa localização das gavetas. Outro aspeto a ter em conta será a disposição dos blocos na gaveta: enquanto que na primeira gaveta os blocos ficaram dispostos o mais direito possível, na segunda gaveta a disposição dos blocos não ficou tão bem conseguida, o que levou a que vários blocos ficassem tortos em relação à orientação desejada. O facto de a imagem ser uma montagem também poderá ter contribuído para o aumento de erro encontrado durante o reconhecimento da imagem.

5.3.2 Teste de quatro gavetas

Analogamente ao capítulo 5.3.1, é abordado o teste efetuado a uma imagem do interior do compartimento do protótipo, contendo quatro gavetas com blocos no seu interior, sendo

que a imagem utilizada é uma montagem obtida através da composição de várias gavetas sozinhas numa só.

Podemos observar na Figura 5.14 (adaptada, como descrito anteriormente) abaixo a localização da área das quatro gavetas:

Figura 5.14 - Visualização das áreas das quatro gavetas

De seguida, mais uma vez, são demonstrados os resultados obtidos (adaptados, como descrito anteriormente), seguidos das tabelas de eficácia correspondentes:

Instituto	Nome	Ano	Número	Tipo	Requisitado?
IVT Errado	****	17	13965	03	Não
IVT Errado	****	17	14497	19	Não
IVT	****	17	14288	03	Não
IVT Errado	****	17	14347	07	Não
IVT Errado	****	17	14290	01	Não
IVT Errado	****	17	13875	11	Não
IVT Errado	****	17	14289	17	Não
IVT Errado	****	17	12634	04	Não
IVT Errado	****	17	14621	05	Não
IVT Errado	****	17	14680	03	Não
IVT Errado	****	17	14231	01	Não
IVT Errado	****	77	14445	18	Não
IVT Errado	****	17	13003	49	Não
IVT Errado	****	17	14703	20	Não
IVT Errado	****	17	12781	05	Não
IVT Errado	****	17	14338	01	Não
IVT	****	17	14345	07	Não
IVT Errado	****	17	13185	08	Não
IVT Errado	****	17	14286	09	Não
IVT Errado	****	17	14286	04	Não

Figura 5.15 - Resultados obtidos através do processo de OCR oriundos dos blocos, visualizados em Excel, da 1ª gaveta

Instituto	Nome	Ano	Número	Tipo	Requisitado?
IVT	****	17	14680	03	Não
IVT	****	17	14338	01	Não
IVT	****	17	14286	09	Não
IVT	****	17	14231	01	Não
IVT Errado	****	17	14445	18	Não
IVT Errado	****	17	14289	17	Não
IVT	****	17	14286	04	Não
IVT Errado	****	17	14497	19	Não
IVT Errado	****	17	13965	03	Não
IVT Errado	****	17	14345	07	Não
IVT Errado	****	7	14347	07	Não
IVT Errado	****	17	14288	03	Não
IVT Errado	****	17	14290	01	Não
IVT Errado	****	17	13003	49	Não
IVT Errado	****	71	4703	20	Não
IVT Errado	****	71	14621	05	Não
IVT Errado	****	71	2781	05	Não
IVT Errado	****	71	2634	04	Não
IVT Errado	****	71	13185	08	Não
IVT Errado	****	71	3875	111	Não

Figura 5.16 - Resultados obtidos através do processo de OCR oriundos dos blocos, visualizados em Excel, da 2ª gaveta

Instituto	Nome	Ano	Número	Tipo	Requisitado?
IVT Errado	****	17	14621	05	Não
IVT Errado	****	17	13003	49	Não
IVT	****	17	14286	09	Não
IVT Errado	****	17	14290	01	Não
IVT	****	17	14231	01	Não
IVT	****	57	12634	04	Não
IVT	****	17	13965	03	Não
IVT	****	17	12781	05	Não
IVT	****	17	14286	04	Não
IVT Errado	****	17	142859	17	Não
IVT	****	17	14288	03	Não
IVT	****	17	14338	01	Não
IVT	****	17	14703	20	Não
IVT Errado	****	17	14445	18	Não
IVT Errado	****	17	13185	08	Não
IVT Errado	****	17	14497	19	Não
IVT Errado	****	17	13875	11	Não
IVT	****	17	14345	07	Não
IVT Errado	****	17	14347	01	Não
IVT Errado	****	17	14680	03	Não

Figura 5.17 - Resultados obtidos através do processo de OCR oriundos dos blocos, visualizados em Excel, da 3ª gaveta

Instituto	Nome	Ano	Número	Tipo	Requisitado?
IVT Errado	****	17	13003	49	Não
IVT	****	17	14345	07	Não
IVT Errado	****	47	14286	09	Não
IVT Errado	****	11	14290	01	Não
IVT Errado	****	77	14445	18	Não
IVT Errado	****	17	12781	05	Não
IVT Errado	****	17	14497	19	Não
IVT Errado	****	17	14288	03	Não
IVT Errado	****	71	3965	03	Não
IVT	****	17	14621	05	Não
IVT Errado	****	17	13185	08	Não
IVT Errado	****	17	14338	01	Não
IVT Errado	****	17	14703	20	Não
IVT Errado	****	17	13875	11	Não
IVT Errado	****	17	12634	04	Não
IVT Errado	****	17	142859	17	Não
IVT Errado	****	17	14286	04	Não
IVT Errado	****	17	14231	01	Não
IVT Errado	****	17	14680	03	Não
IVT Errado	****	17	14347	07	Não

Figura 5.18 - Resultados obtidos através do processo de OCR oriundos dos blocos, visualizados em Excel, da 4ª gaveta

Tabela 5.6 - Eficácia de resultados da primeira gaveta do teste com quatro gavetas

Instituto	Nome	Ano	Número	Tipo	Total
2/20	20/20	20/20	20/20	19/20	81/100
10%	100%	100%	100%	95%	81%

Tabela 5.7 - Eficácia de resultados da segunda gaveta do teste com quatro gavetas

Instituto	Nome	Ano	Número	Tipo	Total
5/20	18/20	13/20	16/20	18/20	70/100
25%	90%	65%	80%	90%	70%

Tabela 5.8 - Eficácia de resultados da terceira gaveta do teste com quatro gavetas

Instituto	Nome	Ano	Número	Tipo	Total
10/20	19/20	19/20	19/20	20/20	87/100
50%	95%	95%	95%	100%	87%

Tabela 5.9 - Eficácia de resultados da quarta gaveta do teste com quatro gavetas

Instituto	Nome	Ano	Número	Tipo	Total
2/20	17/20	16/20	18/20	19/20	72/100
10%	85%	80%	90%	95%	72%

Observando os resultados, notamos que foram mais consistentes entre si, por comparação com os resultados obtidos em 5.3.1. Tendo em conta a Figura 5.14, a hipótese da disposição dos blocos afetar os resultados obtidos ganha força, visto que se obtiveram resultados melhores e mais consistentes em relação aos obtidos em 5.3.1, sendo que os blocos, como se pode observar na Figura 5.14, estão dispostos de uma forma mais ordeira.

De qualquer modo, os resultados obtidos, embora não excelentes, são positivos, e tendo em conta a existência da funcionalidade Proofreader, a tendência será para melhorar com a intervenção do utilizador, o que proporcionará um registo correto da informação, levando a uma gestão dos blocos muito menos atribulada.

5.4 Teste de perceção de profundidade

Este teste irá comparar os resultados obtidos em 5.1 com os resultados obtidos através do teste de uma única gaveta, mas em que a superfície impressa dos blocos não esteja encostada à superfície digitalizadora do *scanner*.

O objetivo é averiguar se o programa produz resultados diferentes, dependendo da aproximação da gaveta à superfície digitalizadora do *scanner*, o que, não sendo uma preocupação em termos de boa implementação do algoritmo, é um cuidado a ter para a boa implementação do *hardware* associado.

Mais uma vez, sendo o processo análogo a todo o processo já descrito durante o capítulo 5, será apenas demonstrada a janela de visualização da área da gaveta, detetada automaticamente, e os resultados obtidos, bem como a tabela de eficácia de que aí advém.

Colocando a gaveta dentro da caixa envolvente, separada do *scanner* por uma distância de aproximadamente 3 cm, obtém-se a seguinte imagem:

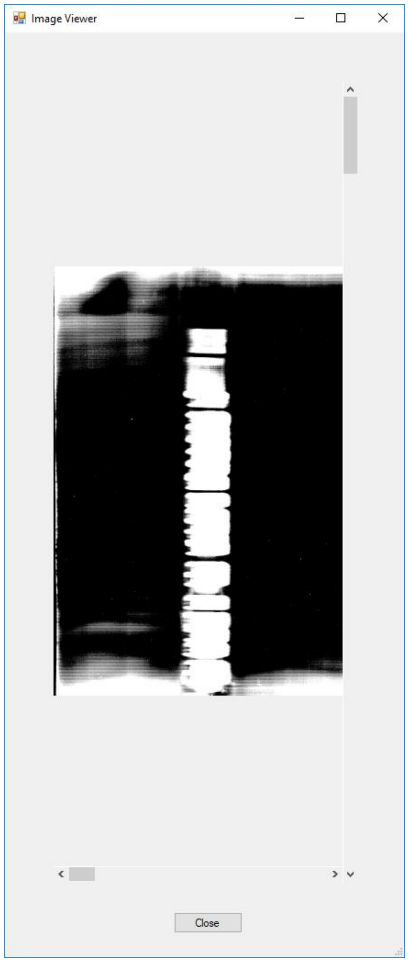


Figura 5.19 - Visualização da área de uma única gaveta a apx. 3 cm de distância do scanner

À primeira vista, facilmente se observa que algo não correu bem: o *scanner*, embora tenha conseguido digitalizar a região geral da gaveta com os blocos, devido à distância entre a mesma e a superfície digitalizadora, foi-lhe impossível digitalizar qualquer informação possível de ser reconhecida. Inclusive, devido à maneira como o algoritmo foi implementado, ele reconhece a área dos *QR codes* e dos blocos em si como sendo na extremidade esquerda da imagem, em vez do centro, onde a gaveta realmente se encontra.

Embora este teste possa parecer algo confuso, e com um resultado possivelmente previsível, ele é necessário tendo em conta o contexto: a ideia original, proposta no capítulo 1, seria a

construção de um aparelho digitalizador, com uma ranhura interior na qual se pudesse inserir gavetas, com dificuldade nula, sendo necessário apenas deslizá-las para dentro e para fora.

No entanto, a gaveta fornecida pelo IVT não tem a mesma altura em todo o seu comprimento: nas suas extremidades frontal e posterior, a gaveta possui uma altura aproximadamente 2.5 cm maior que no comprimento da gaveta. Estes picos nas extremidades impossibilitam assim a possibilidade de as gavetas poderem simplesmente deslizar para dentro e para fora da ranhura, sendo necessário retirar o *scanner* do topo da caixa e voltar a coloca-lo cada vez que se deseja inserir uma ou mais gavetas, algo que, numa versão do protótipo mais avançada, com certeza seria inviável.

5.5 *Teste de orientação angular*

Este teste irá comparar os resultados obtidos em 5.1 com os resultados obtidos através do teste de uma única gaveta, disposta num ângulo não paralelo em relação às paredes da caixa envolvente, tanto para a direita, como para a esquerda.

O objetivo é averiguar se o programa produz resultados diferentes, dependendo do ângulo em que a gaveta se encontra em relação à sua posição vista em 5.1.

Como tem ocorrido até agora, o processo de reconhecimento é análogo ao já descrito no decorrer do capítulo 5, pelo que serão demonstradas as figuras e os resultados associados a este teste, bem como a tabela que daí advém.

Na Figura 5.20 representada abaixo (adaptada, como descrito anteriormente), podemos observar a imagem obtida pelo *scanner* ao digitalizar a gaveta disposta com aproximadamente 25-30º de diferença para com a sua disposição em 5.1, para cada lado:

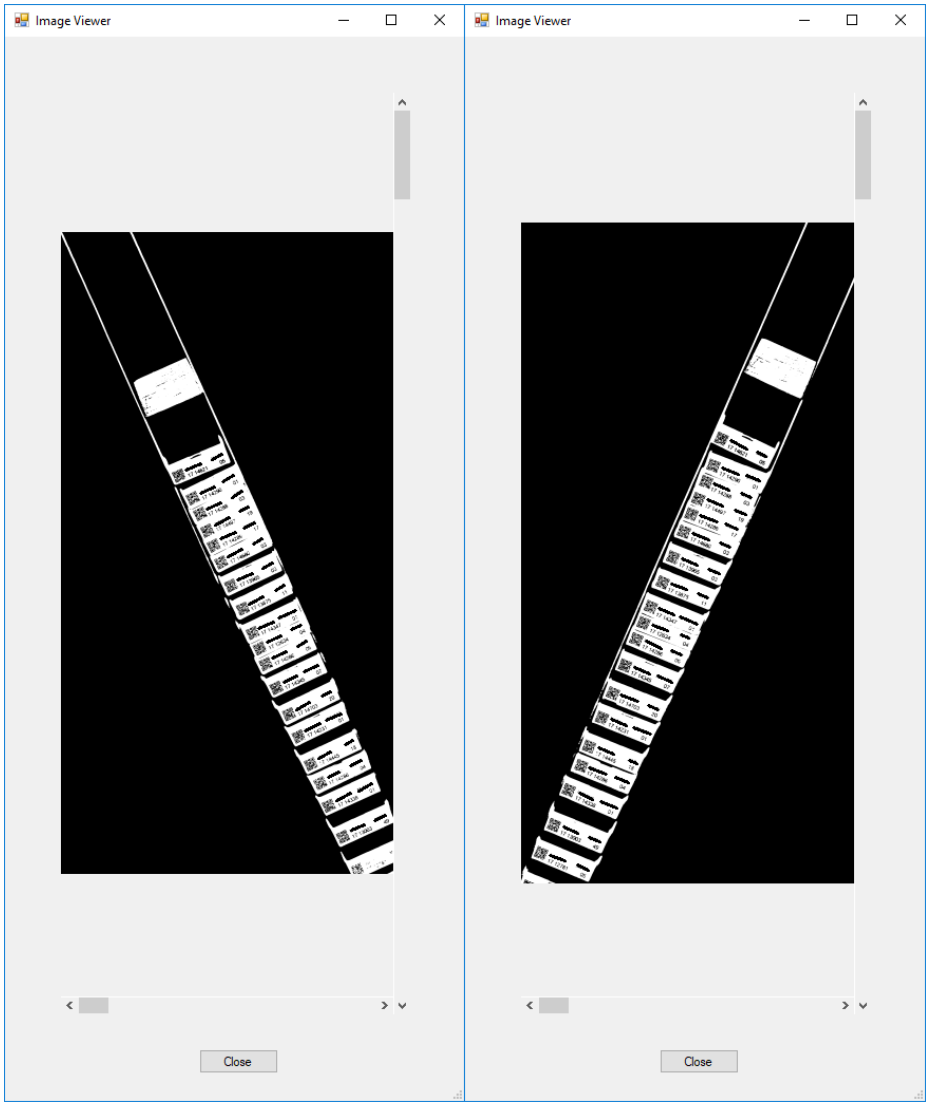


Figura 5.20 - Visualização da área de uma única gaveta rodada para a) a esquerda e para b) a direita

Mais uma vez, considerando o funcionamento do algoritmo implementado, o ângulo das gavetas acima, embora digitalizadas, impossibilita o reconhecimento dos blocos. O método utilizado para a localização dos limites, onde se calculam as variâncias das intensidades na horizontal e na vertical, perde a sua utilidade se a imagem não estiver disposta numa dessas maneiras.

De modo a colmatar essa insuficiência no algoritmo, poder-se-ia implementar uma função que iria detetar o *skew*, ou seja, a inclinação, e posteriormente corrigi-la, rodando a imagem de modo a que esta tenha uma inclinação de 0º, semelhante às condições normais impostas em 5.1. Essa tarefa é, no entanto, de uma dificuldade muito acrescida, que não foi possível implementar neste projeto.

6

Conclusões e Trabalho Futuro

No início desta tese, foi proposta a construção de um sistema, com componente física e informática, que possibilitaria o registo, armazenamento e gestão de blocos histológicos duma forma automatizada em institutos patológicos. Este funcionaria através da adaptação de um *scanner*, para digitalizar as imagens dos blocos, num protótipo a ser futuramente desenvolvido, interligado a um *software* que seria encarregue de tratar do processamento de imagem, reconhecimento ótico de caracteres e centro de gestão dos blocos histológicos.

Chegado a este ponto, o protótipo foi construído e o *software* desenvolvido, bem como se verifica o correto funcionamento da hipótese proposta: o *software* deteta corretamente as gavetas dentro da caixa envolvente, localiza os blocos dentro dela, separa a informação dos blocos com os *QR codes* e reconhece-os com uma percentagem de eficácia geralmente acima dos 80% e nunca menor que 70%, sem correção manual, sendo que com o uso dessa funcionalidade, a percentagem de acerto sobe para praticamente 100%. Tendo tudo isto em conta, pode-se afirmar que existe validação da hipótese proposta.

No entanto, não se trata de um projeto perfeito, havendo ainda muito para melhorar:

- o protótipo é, como o nome indica, ainda muito crude na sua composição. Havendo futuro financiamento para o desenvolvimento comercial deste projeto, seria necessário refazer o desenho do mesmo;
- o algoritmo, embora apresente resultados positivos, ainda pode ser melhorado, nomeadamente na questão da localização, onde ainda trabalha um pouco por

aproximações, o que contribui para um erro sucessivamente maior nos resultados obtidos que dependam de localizações anteriores;

- houve apenas uma gaveta com 20 blocos para testar, pelo que qualquer resultado obtido de testes com várias gavetas não é totalmente fidedigno, assemelhando-se mais a uma simulação que propriamente uma análise propriamente dita;
- o *scanner* utilizado não possui grande perceção de profundidade, pelo que os blocos têm de ser encostados à superfície digitalizadora. O objetivo inicialmente proposto seria semelhante a uma gaveta que entraria e sairia deslizando, o que não acontece, sendo necessário retirar o *scanner* para colocar a gaveta. Descartando a hipótese do IVT efetuar uma mudança em massa dos seus armários e gavetas, será necessário redesenhar a caixa envolvente, ou, caso um eventual orçamento o permita, a aquisição de um *scanner* com melhor perceção de profundidade;
- não foi possível a implementação de um método ou função que pudesse corrigir o *skew* de uma imagem, no entanto, uma hipótese investigada a adotar poderia ser a utilização da transformada de Hough, que possibilita a determinação de linhas numa imagem. Posteriormente, descobrindo o ângulo de inclinação, tornar-se-ia possível a correção de ângulos dentro de uma imagem, não só da gaveta, como também dos blocos, cuja disposição muitas vezes fica torta em relação à gaveta, dificultando o reconhecimento com sucesso. Conseguindo realizar esta implementação, o programa ganharia uma poderosa ferramenta para auxiliar o reconhecimento;
- durante a realização do *software* principal, efetuou-se inesperadamente uma atualização no *software* do Office, o que causou efeitos inesperados na normal execução do programa, nomeadamente a incompatibilização das funcionalidades da programação do Excel: tornou-se impossível gravar automaticamente os resultados obtidos num ficheiro Excel, bem como aceder a um ficheiro Excel através do Block Manager. Este incidente torna óbvio como necessária a atenção à estabilidade dos *softwares* adicionais utilizados na construção do programa, bem como a compatibilidade do programa para com várias versões desses mesmos *softwares*. No entanto, é de realçar que o *software* principal é construído com atenção à compatibilidade com computadores diferentes, tendo sido cautelosamente escolhida uma abordagem abrangente à escolha de caminhos de ficheiros e pastas a aceder;
- muitos dos blocos já se encontravam em condições mais degradadas, o que contribuiu também para um aumento do erro nos resultados obtidos, nomeadamen-

te, os *QR codes*, que tiveram de ser reconstruídos no computador, impressos e colados nos blocos de modo a conseguirem ser reconhecidos, de outro modo nenhum conseguiria ser reconhecido com sucesso.

Em relação a trabalho futuro, é uma incógnita: a realização do projeto é do interesse do IVT, cujo sistema atual em nada é automatizado, pelo que a implementação deste projeto nas suas instalações poderia aumentar bastante a sua eficiência. No entanto, a existência do FINA põe uma grande questão: seria possível desenvolver e comercializar este projeto, competindo com o FINA, evitando possíveis ações judiciais ou até conseguir patentes? Nesse aspeto, a possibilidade de trabalho futuro apresenta-se como uma possibilidade remota, no entanto, no foro privado, há imenso espaço para melhoramentos no sistema, maioritariamente no protótipo, mas também no *software*.

7

Referências

[1] "Significado / definição de biopsia no Dicionário Priberam da Língua Portuguesa." [Online]. Available: https://www.priberam.pt/DLPO/biopsia. [Accessed: 02-Mar-2017].

[2] L. Biosystems, "An Introduction to Specimen Processing," no. May, pp. 1–5, 2011.

[3] "Biopsy Cassettes - Product: Leica Biosystems." [Online]. Available: http://www.leicabiosystems.com/histology-consumables/cassettes/biopsy-cassettes/products/biopsy-cassettes/. [Accessed: 02-Mar-2017].

[4] "Leica SM2010 R - Produto: Leica Biosystems." [Online]. Available: http://www.leicabiosystems.com/pt/equipamento-de-histologia/microtomos-deslizantes/detalhes/product/leica-sm2010-r/. [Accessed: 02-Mar-2017].

[5] "Legislação / Manual de Boas Práticas de Anatomia Patológica." [Online]. Available: https://www.ordemdosmedicos.pt/?lop=conteudo&op=c058f544c737782deacefa532d9add4c&id=020c8bfac8de160d4c5543b96d1fdede. [Accessed: 02-Mar-2017].

[6] "Epson Perfection V19 Scanner | Photo | Scanners | For Home | Epson US." [Online]. Available: https://epson.com/For-Home/Scanners/Photo/Epson-Perfection-V19-Scanner/p/B11B231201. [Accessed: 02-Mar-2017].

[7] "Windows Image Acquisition (WIA) | Microsoft Docs." [Online]. Available:

https://docs.microsoft.com/pt-pt/windows/desktop/wia/-wia-startpage. [Accessed: 24-Oct-2018].

[8] The TWAIN Working Group, "Linking Images With Applications," 2011.

[9] The TWAIN Working Group, "The Origin of TWAIN," 2009.

[10] P. Translations, "ISIS ® An Image and Scanner Interface Specification ISIS An Image and Scanner Interface Specification."

[11] SANE, "SANE Standard Version 1.06 - Programmer's Documentation," 2008.

[12] "Sane Frequently Asked Questions." [Online]. Available: https://ljm.home.xs4all.nl/SANE-faq.html#7. [Accessed: 13-Nov-2018].

[13] "About Asprise: 15+ years of offering software SDK library API for OCR, Java/.NET document scanner scanning components and PDF/TIFF imaging toolkits." [Online]. Available: http://asprise.com/ocr-document-scanning-java.net/library-api-about.html. [Accessed: 02-Mar-2017].

[14] "Asprise Java OCR SDK - royalty-free API library with source code examples converting images to word or searchable PDF by extracting text." [Online]. Available: http://asprise.com/royalty-free-library/ocr-api-for-java-csharp-vb.net.html. [Accessed: 02-Mar-2017].

[15] "Demos of Asprise C# .NET OCR SDK - royalty-free API library with source code examples converting images to word or searchable PDF by extracting text." [Online]. Available: http://asprise.com/royalty-free-library/c%23-sharp.net-ocr-source-code-examples-demos.html. [Accessed: 02-Mar-2017].

[16] "Key Facts About ABBYY." [Online]. Available: https://www.abbyy.com/en-eu/company/key-facts/. [Accessed: 02-Mar-2017].

[17] "PDF software with OCR for Windows." [Online]. Available: https://www.abbyy.com/en-eu/finereader/in-details/. [Accessed: 02-Mar-2017].

[18] "PDF converter to Excel and Word with OCR." [Online]. Available: https://www.abbyy.com/en-eu/finereader/convert/. [Accessed: 02-Mar-2017].

[19] "Why FineReader." [Online]. Available: https://www.abbyy.com/en-eu/finereader/why-finereader/. [Accessed: 02-Mar-2017].

[20] R. Smith, "An overview of the tesseract OCR engine," *Proc. Int. Conf. Doc. Anal. Recognition, ICDAR*, vol. 2, pp. 629–633, 2007.

[21] R. Smith and Z. Podobny, "Tesseract Open Source OCR Engine," 2015.

[Online]. Available: https://github.com/tesseract-ocr/tesseract.

[22] "Tesseract: an Open-Source Optical Character Recognition Engine | Linux Journal." [Online]. Available: http://www.linuxjournal.com/article/9676. [Accessed: 02-Mar-2017].

[23] J. Muccigrosso, "Tesseract FAQ." [Online]. Available: https://github.com/tesseract-ocr/tesseract/wiki/FAQ.

[24] "Dreampath Diagnostics - Our solutions." [Online]. Available: http://www.dreampathdx.net/our-solutions. [Accessed: 02-Mar-2017].

[25] S. a. Instrumentos Cientificios, "REF. APCS01. Escáner para bloques de parafina. Modelo: FINA. Marca: DREAMPATH." [Online]. Available: https://cdn.icsa.es:8080/wp-content/uploads/2016/07/160122-Descripcion-tecnica-escaner-Fina.pdf. [Accessed: 02-Mar-2017].

[26] R. C. Gonzalez and R. E. Woods, "Digital Image Processing Third Edition," 2008.

[27] J. M. Fonseca, "Acondicionamento de imagem." .

[28] J. M. Fonseca, "Objectos Isolados." pp. 1–14.

[29] L. Eikvil, "Optical character recognition," *Citeseer. Ist. Psu. Edu/142042. Html*, vol. 3, no. 1, pp. 4956–4958, 1993.

[30] P. Barroso, J. Amaral, A. Mora, J. M. Fonseca, and A. Steiger-Garção, "A Quadtree Based Vehicles Recognition System," *4th WSEAS Int. Conf. Opt. Photonics, Lasers Imaging (ICOPLI 2004)*, vol. 1, pp. 12–16, 2004.

[31] B. Enyedi, L. Konyha, and K. Fazekas, "Real time number plate localization algorithms," *J. Electr. Eng.*, vol. 57, no. 2, pp. 69–77, 2006.

[32] J. M. Fonseca, "Binarização da imagem," pp. 1–11.

[33] R. P. Van Heerden and E. C. Botha, "Optimization of vehicle licence plate segmentation and symbol recognition."

[34] R. Thomas, "Tessnet2 a .NET 2.0 Open Source OCR assembly using Tesseract engine." [Online]. Available: http://www.pixel-technology.com/freeware/tessnet2/. [Accessed: 24-Sep-2018].

[35] A. Lima, "Como escanear documentos com o C#? (digitalização de documentos) - André Alves de Lima," 2015. [Online]. Available: http://www.andrealveslima.com.br/blog/index.php/2015/12/16/como-escanear-documentos-com-o-c-digitalizacao-de-documentos/. [Accessed: 22-Jan-2019].

[36] C. Delgado, "Creating a scanning application in Winforms with C# | Our

Code World," 2017. [Online]. Available: https://ourcodeworld.com/articles/read/382/creating-a-scanning-application-in-winforms-with-csharp. [Accessed: 22-Jan-2019].

[37] Microsoft, "How to: Access Office Interop Objects by Using Visual C# Features - C# Programming Guide | Microsoft Docs," 2015. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/interop/how-to-access-office-onterop-objects. [Accessed: 22-Jan-2019].

[38] Microsoft, "How to automate Microsoft Excel from Microsoft Visual C#.NET," 2018. [Online]. Available: https://support.microsoft.com/pt-pt/help/302084/how-to-automate-microsoft-excel-from-microsoft-visual-c-net. [Accessed: 22-Jan-2019].

[39] Bospear, "C# Excel Tutorials - YouTube," 2017. [Online]. Available: https://www.youtube.com/playlist?list=PLKEDZb_wChLesXK9M2zmFy8ybgbDHkT9l. [Accessed: 22-Jan-2019].

[40] LearnCoding, "How To Edit Excel Worksheet Using C# -Part 7- (Write Data to Specific Cells) - YouTube," 2016. [Online]. Available: https://www.youtube.com/watch?v=EjES2ADnoQU&list=PLEdObNxHtDHI-EBwVmLLrfdsMp0hsydWV&index=6. [Accessed: 22-Jan-2019].

[41] Bsargos, "How to Integrate Excel in a Windows Form Application using the WebBrowser," 2006. [Online]. Available: https://www.codeproject.com/articles/15760/how-to-integrate-excel-in-a-windows-form-applicati. [Accessed: 22-Jan-2019].

8

Anexos

8.1 Fluxogramas

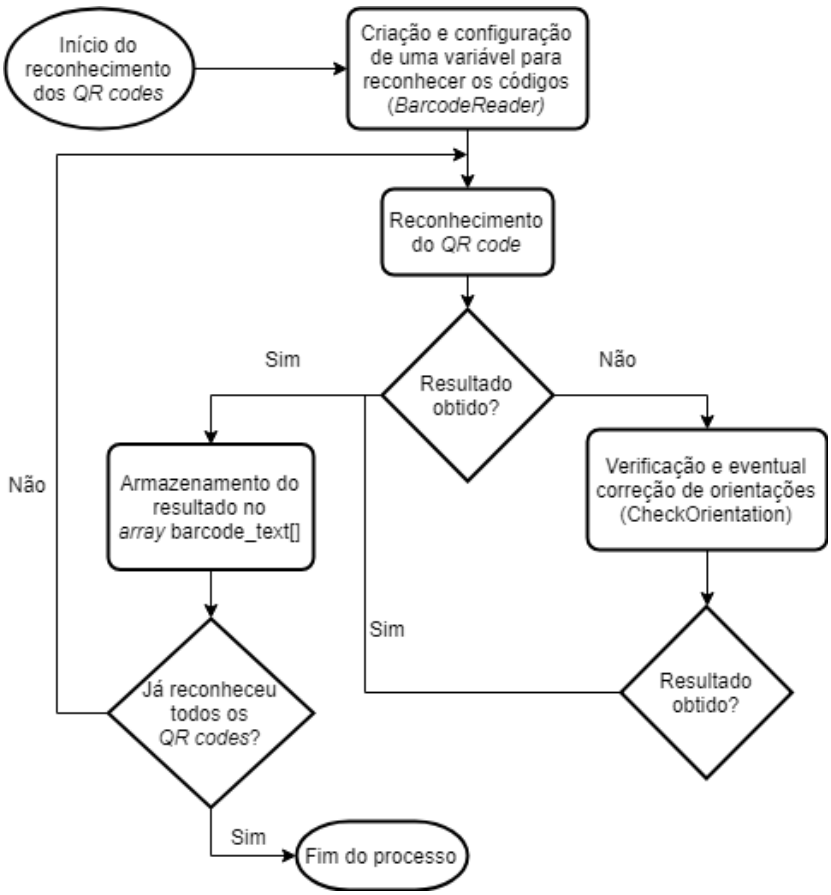


Figura 8.1 – Fluxograma referente ao funcionamento da função "Decode_QR"

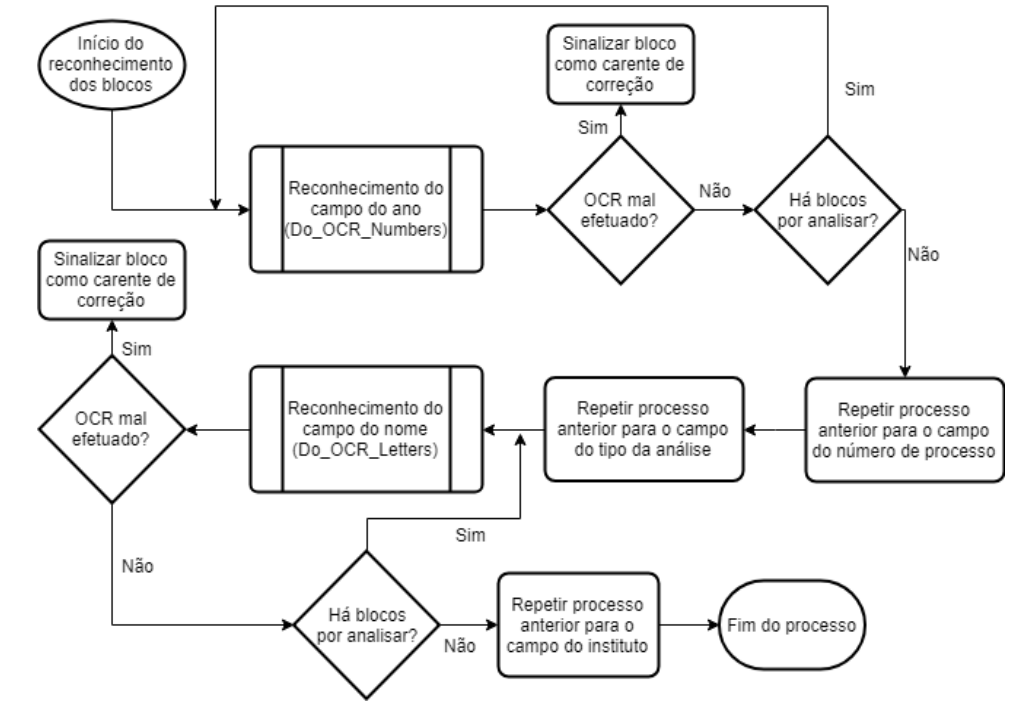


Figura 8.2 – Fluxograma referente ao funcionamento da função "Perform_OCR"

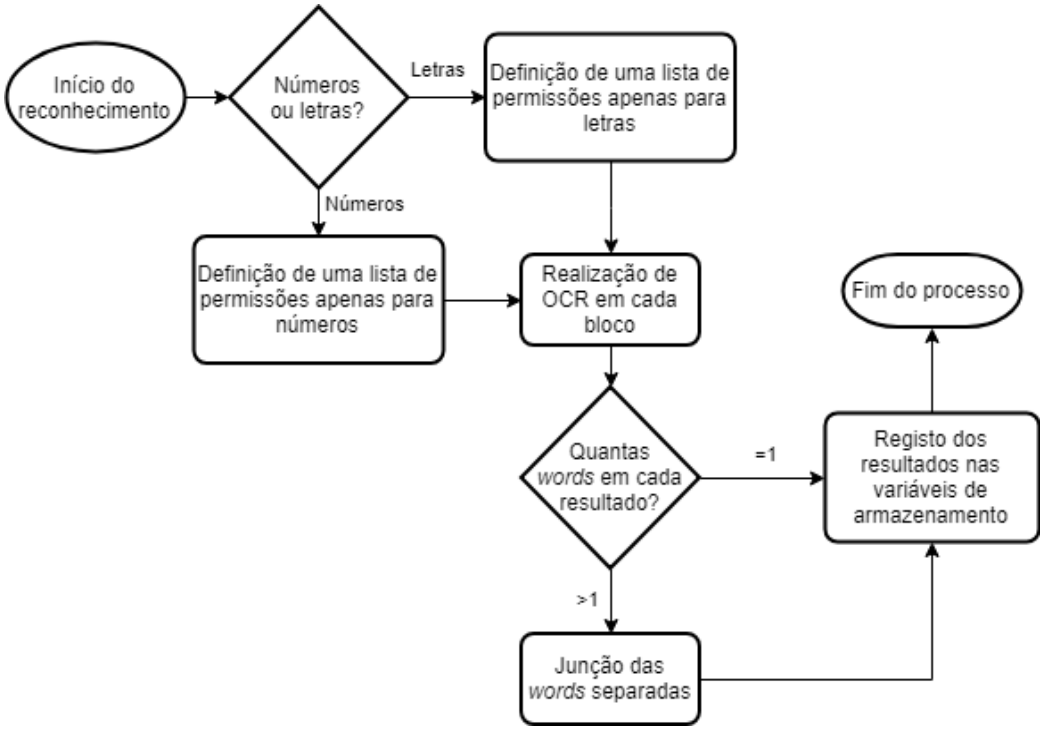


Figura 8.3 – Fluxograma referente ao funcionamento das funções "Do_OCR_Numbers" e "Do_OCR_Letters"

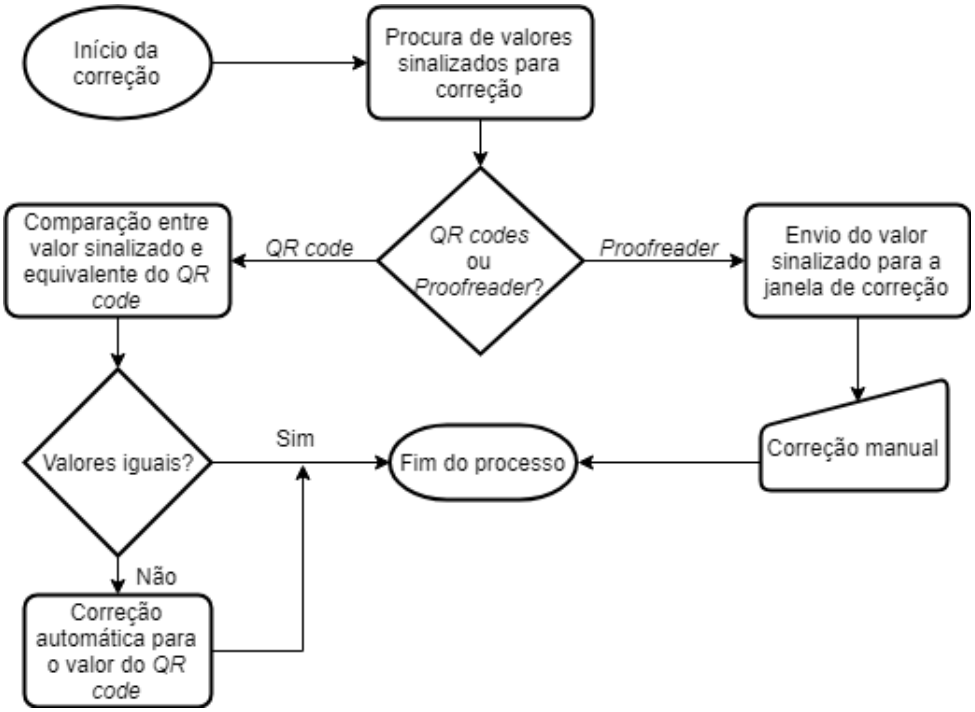


Figura 8.4 – Fluxograma referente ao funcionamento das funções "QR_Correction" e "Proofreader_Correction"

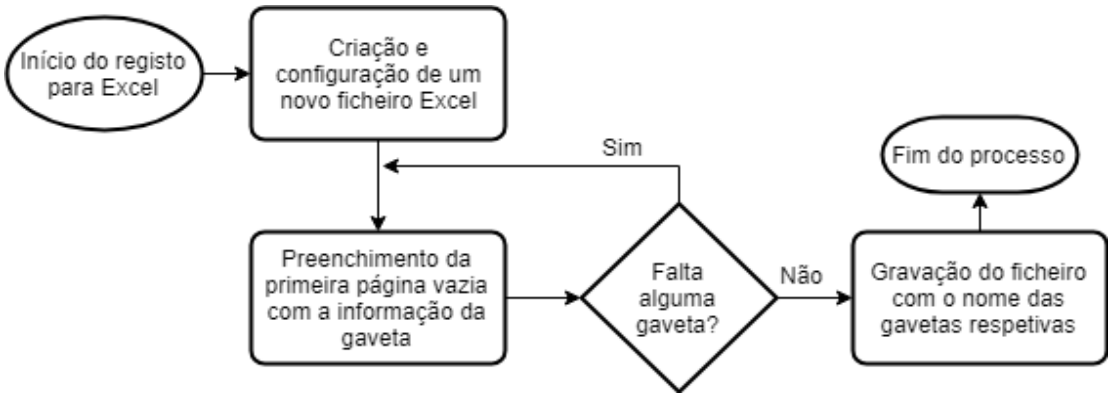


Figura 8.5 – Fluxograma referente ao funcionamento da função "Excel_Save"

8.2 *Código implementado*

8.2.1 WIAClass

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using WIA;
using System.Runtime.InteropServices;
using System.Drawing;
using System.IO;
using Emgu.CV;
using Emgu.CV.UI;
using Emgu.CV.Structure;
using Emgu.CV.CvEnum;

namespace Block_Manager_2018
{
    class WIAClass
    {
        public void AdjustScannerSettings(IItem scannnerItem, int
scanResolutionDPI, int scanStartLeftPixel, int scanStartTopPixel,
                    int scanWidthPixels, int scanHeightPixels, int
brightnessPercents, int contrastPercents, int colorMode)
        {
            const string WIA_SCAN_COLOR_MODE = "6146";
            const string WIA_HORIZONTAL_SCAN_RESOLUTION_DPI = "6147";
            const string WIA_VERTICAL_SCAN_RESOLUTION_DPI = "6148";
            const string WIA_HORIZONTAL_SCAN_START_PIXEL = "6149";
            const string WIA_VERTICAL_SCAN_START_PIXEL = "6150";
            const string WIA_HORIZONTAL_SCAN_SIZE_PIXELS = "6151";
            const string WIA_VERTICAL_SCAN_SIZE_PIXELS = "6152";
            const string WIA_SCAN_BRIGHTNESS_PERCENTS = "6154";
            const string WIA_SCAN_CONTRAST_PERCENTS = "6155";
            SetWIAProperty(scannnerItem.Properties,
WIA_HORIZONTAL_SCAN_RESOLUTION_DPI, scanResolutionDPI);
            SetWIAProperty(scannnerItem.Properties,
WIA_VERTICAL_SCAN_RESOLUTION_DPI, scanResolutionDPI);
            SetWIAProperty(scannnerItem.Properties,
WIA_HORIZONTAL_SCAN_START_PIXEL, scanStartLeftPixel);
            SetWIAProperty(scannnerItem.Properties,
WIA_VERTICAL_SCAN_START_PIXEL, scanStartTopPixel);
            SetWIAProperty(scannnerItem.Properties,
WIA_HORIZONTAL_SCAN_SIZE_PIXELS, scanWidthPixels);
            SetWIAProperty(scannnerItem.Properties,
WIA_VERTICAL_SCAN_SIZE_PIXELS, scanHeightPixels);
            SetWIAProperty(scannnerItem.Properties,
WIA_SCAN_BRIGHTNESS_PERCENTS, brightnessPercents);
            SetWIAProperty(scannnerItem.Properties,
WIA_SCAN_CONTRAST_PERCENTS, contrastPercents);
            SetWIAProperty(scannnerItem.Properties,
WIA_SCAN_COLOR_MODE, colorMode);
```

```
        }

        private static void SetWIAProperty(IProperties properties, ob-
ject propName, object propValue)
        {
            Property prop = properties.get_Item(ref propName);
            prop.set_Value(ref propValue);
        }

        public void ShowErrorCode(COMException err)
        {
            uint errorCode = (uint)err.ErrorCode;

            switch (errorCode)
            {
                case 0x80210006:
                    MessageBox.Show("The device is busy. Close any
apps that are using this device or wait for it to finish and then try
again.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    break;

                case 0x80210016:
                    MessageBox.Show("One or more of the device's cover
is open.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    break;

                case 0x8021000A:
                    MessageBox.Show("Communication with the WIA device
failed. Make sure that the device is powered on and connected to the
PC. If the problem persists, disconnect and reconnect the device.",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    break;

                case 0x8021000D:
                    MessageBox.Show("The device is locked. Close any
apps that are using this device or wait for it to finish and then try
again.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    break;

                case 0x8021000E:
                    MessageBox.Show("The device driver threw an excep-
tion.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    break;

                case 0x80210001:
                    MessageBox.Show("An unknown error has occurred
with the WIA device.", "Error", MessageBoxButtons.OK, MessageBoxI-
con.Error);
                    break;

                case 0x8021000C:
                    MessageBox.Show("There is an incorrect setting on
the WIA device.", "Error", MessageBoxButtons.OK, MessageBoxI-
con.Error);
                    break;

                case 0x8021000B:
                    MessageBox.Show("The device doesn't support this
```

```
command.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    break;

                case 0x8021000F:
                    MessageBox.Show("The response from the driver is
invalid.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    break;

                case 0x80210009:
                    MessageBox.Show("The WIA device was deleted. It's
no longer available.", "Error", MessageBoxButtons.OK, MessageBoxI-
con.Error);
                    break;

                case 0x80210017:
                    MessageBox.Show("The scanner's lamp is off.", "Er-
ror", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    break;

                case 0x80210021:
                    MessageBox.Show("A scan job was interrupted be-
cause an Imprinter/Endorser item reached the maximum valid value for
WIA_IPS_PRINTER_ENDORSER_COUNTER, and was reset to 0.", "Error", Mes-
sageBoxButtons.OK, MessageBoxIcon.Error);
                    break;

                case 0x80210020:
                    MessageBox.Show("A scan error occurred because of
a multiple page feed condition.", "Error", MessageBoxButtons.OK, Mes-
sageBoxIcon.Error);
                    break;

                case 0x80210005:
                    MessageBox.Show("The device is offline. Make sure
the device is powered on and connected to the PC.", "Error", Message-
BoxButtons.OK, MessageBoxIcon.Error);
                    break;

                case 0x80210007:
                    MessageBox.Show("The device is warming up.", "Er-
ror", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    break;

                case 0x80210008:
                    MessageBox.Show("There is a problem with the WIA
device. Make sure that the device is turned on, online, and any cables
are properly connected.", "Error", MessageBoxButtons.OK, MessageBoxI-
con.Error);
                    break;

                case 0x80210015:
                    MessageBox.Show("No scanner device was found. Make
sure the device is online, connected to the PC, and has the correct
driver installed on the PC.", "Error", MessageBoxButtons.OK, Message-
BoxIcon.Error);
                    break;

                case 0x80210064:
```

```
                    MessageBox.Show("The scanning process has been
cancelled.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    break;

            }
        }

        public void MainFunction()
        {
            String path = Application.StartupPath + "\\Images\\";
//Startup path que acede imediatamente à pasta bin/debug/Images

            try
            {
                ////////////////////////////////////////
                //      DECLARAÇAO DE VARIAVEIS       //
                ////////////////////////////////////////

                var dialog = new WIA.CommonDialogClass();

                WIA.Device scanner = dia-
log.ShowSelectDevice(WiaDeviceType.ScannerDeviceType, true, false);
//Escolher o scanner que quer utilizar, pode haver outro dispositivo
ligado ao computador, é sempre bom dar a escolha
                WIA.Item scannerItem = scanner.Items[1];
//Cria um WIA.Item do scanner, de modo a poder utilizar e configurar
os valores inerentes ao scanner

                var h_extent =
(int)scannerItem.Properties.get_Item("6151").get_Value();
//Vai buscar o valor de largura default do scanner atraves das propri-
edades do scannerItem
                var v_extent =
(int)scannerItem.Properties.get_Item("6152").get_Value();
//Vai buscar o valor de altura default do scanner atraves das proprie-
dades do scannerItem
                int dpi = 600;
//Define o valor de dpi, pode ser mudado pelo developer mas nao pelo
utilizador, manter?
                int dpi_factor = dpi / 100;
//Divide o valor de dpi por 100 para multiplicar pela largura e altura
do scanner, de modo a obter altura e largura da imagem
                int h_res = h_extent * dpi_factor;
//Valor de largura da imagem
                int v_res = v_extent * dpi_factor;
//Valor de altura da imagem


                ////////////////////////////////////////
                //  AJUSTAMENTO DOS SETTINGS DO SCANNER //
                ////////////////////////////////////////

                AdjustScannerSettings(scannerItem, dpi, 0, 0, h_res,
v_res, 0, 0, 2);                    //Atribui novos valores às proprie-
dades do scanner, de acordo com a vontade do developer
                var scannedImage = dialog.ShowTransfer(scannerItem,
WIA.FormatID.wiaFormatPNG) as WIA.ImageFile;        //Adquire imagem
como WIA.ImageFile diretamente do scanner, com os parametros estabele-
```

```
cidos acima, mostrando apenas a barra de progresso


                ////////////////////////////////////////
                //      SCAN E CRIAÇAO DA IMAGEM        //
                ////////////////////////////////////////

                if (scannedImage != null)       //Caso o scan tenha
tido sucesso
                {
                    var imageBytes = (by-
te[])scannedImage.FileData.get_BinaryData(); // Conversao do ImageFile
para um array de bytes

                    MemoryStream ms = new MemoryStream(imageBytes);
//Passa o array de bytes para uma memory stream
                    Image imageFS = Image.FromStream(ms);
//Converte a memory stream numa imagem

                    Bitmap bmpImage = new Bitmap(imageFS);
//Cria um bitmap atraves da imagem
                    Image<Bgr, byte> image = new Image<Bgr, by-
te>(bmpImage);    //Cria uma imagem BGR atraves do bitmap
                    image = image.Rotate(180, new Bgr(0, 0, 0), fal-
se);          //Roda 180 graus porque a orientaçao do scanner com a ga-
veta é inversa
                    ImageViewer.Show_Image(image);
                    bmpImage = image.ToBitmap();
//Volta a converter para bitmap e reinsere o dpi de 600, porque ao
converter para Image<Bgr, byte>
                    bmpImage.SetResolution(dpi, dpi);
//o dpi foi reduzido para 96, valor arbitrário de dpi tido em imagens
criadas a computador

                    if (System.IO.File.Exists(path + "scanwia.png"))
                    {
                        bmpImage.Save(path + "scanwia_" + DateTi-
me.Now.ToString("ddMMyyyy_HHmmss") + ".png");        //Se já existir
imagem, cria-se nova, com marca de data
                    }
                    else
                    {
                        bmpImage.Save(path + "scanwia.png");
//Se nao, cria-se uma
                    }
                }
            }
            catch (COMException err)                    // Em caso de
erro, faz switch ao código de erro, e mostra ao utilizador o erro cor-
respondente
            {
                ShowErrorCode(err);
            }
        }

    }
}
```

8.2.2 RecognitionClass (apenas MainFunction)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Drawing;
using System.Windows.Forms;
using Emgu.CV;
using Emgu.CV.UI;
using Emgu.CV.Structure;
using Emgu.CV.CvEnum;
using tessnet2;
using ZXing;
using ZXing.Common;
using ZXing.QrCode;
using System.Runtime.InteropServices;

namespace Block_Manager_2018
{
    public class RecognitionClass
    {

        public class User_Info
        {
            public string place;
            public string name;
            public int year;
            public int file_num;
            public int analysis_type;
            public int[] confidence = new int[5];
        }

        public void MainFunction()
        {

            ////////////////////////////////////////
            //      DECLARAÇAO DE VARIAVEIS       //
            ////////////////////////////////////////

            OpenFileDialog dlg = new OpenFileDialog();      //Abre fi-
le dialog para o utilizador escolher o ficheiro que quer fazer OCR
            DialogResult rslt = dlg.ShowDialog();           //Cria uma
variável rslt, para controlar o evento de saída do dialog

            if (rslt != DialogResult.OK)                    //Se não
foi escolhido um ficheiro para abrir antes de se fechar o dialog, o
programa sai da função, de modo a impedir quaisquer problemas futuros
            {
                return;
            }

            var imageOCR = new Bitmap(dlg.OpenFile());      //Cria uma
variável Bitmap que vai receber o ficheiro escolhido pelo utilizador

            float k = imageOCR.HorizontalResolution;
```

```
//Declaracao das variaveis k e dpi, que recebem, respetivamente, a re-
solucao horizontal da imagem em float, e a sua conversao para int
            int dpi = Convert.ToInt32(k);

            int i, j, x, y;

            Image<Bgr, byte> imageEmgu = new Image<Bgr, by-
te>(imageOCR);    //Conversao da imagem imageOCR de var para uma vari-
avel Image<Bgr, byte> de nome imageEmgu, de modo a poder tratar a ima-
gem
            Image<Bgr, byte> imageCopy = imageEmgu.Copy();
//Criacao de tres variaveis imageCopy, para poder trabalhar a imagem
com segurança
            Image<Bgr, byte> imageCopy2 = null;
            Image<Gray, byte> imageCopy3 = null;
//Esta é uma Image Gray em vez de Bgr, de modo a facilitar o seu pro-
cesso
            Image<Bgr, byte> imageQR = null;
//Criaçao de uma imagem imageQR que vai receber apenas a "fatia" dos
codigos QR


            ////////////////////////////////////////////////
            //     AVERIGUACAO DOS DETALHES DAS GAVETAS    //
            ////////////////////////////////////////////////

            DrawerNumberForm drawer_num_form = new DrawerNumberForm();
//Form simples, cuja única função é receber do utilizador e enviar ao
programa o número de gavetas a reconhecer
            drawer_num_form.ShowDialog();
//Devido a restrições de hardware, o número máximo de gavetas que o
form vai receber são 6
            int drawer_num = drawer_num_form.GetNumber();

            string[] drawer_name = new string[drawer_num];
//Vai receber e enviar o nome de cada gaveta
            DrawerNameForm drawer_name_form = new DrawerNameForm();

            for (i = 0; i < drawer_num; i++)
            {
                drawer_name_form.ShowDialog();
                drawer_name[i] = drawer_name_form.GetName();
//Enquanto houverem gavetas sem nome, o form vai aparecendo para que o
utilizador insira os nomes
            }


            ////////////////////////////////////////////
            //      PRE-PROCESSAMENTO DA IMAGEM      //
            ////////////////////////////////////////////

            ImageClass.Mean_NxN(imageCopy, imageEmgu, 3);       //Faz
o smoothing da imagem com um cluster 3x3 - pré-processamento, tratando
a imagem de copia usando a imagem original como pontos de referencia
            ImageClass.ConvertToBW_Otsu(imageCopy);
//Conversão Otsu para preto e branco

            Rectangle[] drawer_area = new Rectangle[drawer_num];
```

```
            drawer_area = ImageClass.locateLimits(imageCopy, dpi,
drawer_num);    //Chama a função locateLimits e retorna o Rectangle
com as coordenadas das gavetas

            Rectangle[] qr_area = new Rectangle[drawer_num];
//Declaram-se dois rects: qr_area e blocks_area, que vão receber as
coordenadas de drawer_area e que serão posteriormente

//modificados, com valores conseguidos através de analise anterior à
imagem
            for (i = 0; i < qr_area.Length; i++)
            {
                qr_area[i] = drawer_area[i];
                qr_area[i].Width = (dpi / 11) * 4;
            }

            Rectangle[] blocks_area = new Rectangle[drawer_num];

            for (i = 0; i < qr_area.Length; i++)
            {
                blocks_area[i] = drawer_area[i];
                blocks_area[i].X += (dpi / 3);
                blocks_area[i].Width -= (dpi / 3);

                if(blocks_area[i].Width > 500)
//Por experiência, é sabido que o valor de largura de um bloco geral-
mente não excede os 466
                {
//Desse modo, se houver um bloco muito largo, a sua largura é restrita
em 466
                    blocks_area[i].Width = 466;
                }

            }

            ImageViewer.Show_Image(imageCopy.Copy(qr_area[0]));
            ImageViewer.Show_Image(imageCopy.Copy(blocks_area[0]));


            ////////////////////////////////////////
            //   DECLARAÇAO DOS ARRAYS DOS BLOCOS   //
            ////////////////////////////////////////

            Rectangle[] blocks;   //Declaração do vector de rects
"blocks", que vai receber em cada posição as coordenadas de cada bloco
na imagem

            Rectangle[] blocks_letters;       //O mesmo que acima, mas
recebe apenas a parte superior do bloco
            Rectangle[] blocks_numbers;       //O mesmo que acima, mas
recebe apenas a parte inferior do bloco

            Rectangle[] blocks_letters_left; //Recebe a metade direita
de blocks_letters, i.e. o nome da pessoa, efetivamente
            Rectangle[] blocks_letters_right; //Recebe a metade direi-
ta de blocks_letters, i.e. o nome da pessoa, efetivamente

            Rectangle[] blocks_num_left;      //Recebe a parte esquer-
```

```
da de blocks_numbers, i.e. o ano
            Rectangle[] blocks_num_mid;       //Recebe a parte do meio
de blocks_numbers, i.e. o número de processo
            Rectangle[] blocks_num_right;     //Recebe a parte direita
de blocks_numbers, i.e. o número relativo ao tipo de análise

            Rectangle[] qr_blocks;      //Recebe as coordenadas de ca-
da código QR

            int[] year_end;             //Recebe a coordenada X que
coincide com o final de impressão do ano em cada bloco
            string[] barcode_text;      //Recebe o texto recolhido dos
códigos QR

            User_Info[] blocks_info;            //Array de objectos
da classe User_Info, criado com o número de blocos existentes, tendo
cada objeto
                                                //a informacao de ca-
da bloco registada em 6 campos diferentes e simples de diferenciar

            User_Info[][] drawer_info;          //Array de objetos da
classe User_Info, criado com o número de gavetas existentes, e que vai
recolher
                                                //todas as informa-
ções de blocks_info para cada gaveta.

            drawer_info = new User_Info[drawer_num][];

            for (y = 0; y < drawer_num; y++)
            {
                drawer_info[y] = new User_Info[50];
//Inicialização dos drawer_num elementos de drawer_info, sendo que ca-
da um será um array de objetos da classe User_Info
            }                                                   //Como neste
momento o programa ainda não sabe quantos blocos vai reconhecer, são
inicializadas 50 posições de prevenção
                                                                //Foram efetu-
adas medidas no comprimento da gaveta e dos blocos, o que deu uma mé-
dia de 45-50 blocos possíveis de serem inseridos,
                                                                //dependendo
da existência ou não de outros objetos dentro da gaveta

            for (y = 0; y < drawer_num; y++)
            {
                for (x = 0; x < 50; x++)
                {
                    drawer_info[y][x] = new User_Info();
//Inicialização de cada objeto de classe User_Info existentes em
drawer_info
                }
            }


            ////////////////////////////////////////
            //   OCR E LOCALIZAÇAO DOS BLOCOS E QR  //
            ////////////////////////////////////////

            for (i = 0; i < drawer_num; i++)
```

```
            {

                imageQR = imageCopy.Copy(qr_area[i]);
//Definidas os rects qr_area e blocks_area, este são utilizados para
fazer recortes da imagem original para imageQR e imageCopy2
                imageCopy2 = imageCopy.Copy(blocks_area[i]);
                imageCopy3 = imageCopy2.Convert<Gray, byte>();
//Converte imageCopy2 para uma imagem em escala de cinzento, e envia
para imageCopy3

                blocks = ImageClass.locateBlocks(imageCopy2, dpi);
//Declaração do vector de rects "blocks", que vai receber em cada po-
sição as coordenadas de cada bloco na imagem

                blocks_letters = new Rectangle[blocks.Length];
//O mesmo que acima, mas recebe apenas a parte superior do bloco
                blocks_numbers = new Rectangle[blocks.Length];
//O mesmo que acima, mas recebe apenas a parte inferior do bloco

                blocks_letters_left = new Rectangle[blocks.Length];
//Recebe a metade direita de blocks_letters, i.e. o nome da pessoa,
efetivamente
                blocks_letters_right = new Rectangle[blocks.Length];
//Recebe a metade direita de blocks_letters, i.e. o nome da pessoa,
efetivamente

                blocks_num_left = new Rectangle[blocks.Length];
//Recebe a parte esquerda de blocks_numbers, i.e. o ano
                blocks_num_mid = new Rectangle[blocks.Length];
//Recebe a parte do meio de blocks_numbers, i.e. o número de processo
                blocks_num_right = new Rectangle[blocks.Length];
//Recebe a parte direita de blocks_numbers, i.e. o número relativo ao
tipo de análise

                qr_blocks = ImageClass.locateQR(imageQR, dpi);
//Recebe as coordenadas de cada código QR

                year_end = new int[blocks.Length];
//Vai receber, para cada bloco, o número da coluna onde acaba de ser
"escrito" o segundo algarismo referente ao ano

                for (j = 0; j < qr_blocks.Length; j++)
//Como as coordenadas qr_blocks foram conseguidas dentro de imageQR,
que tem apenas a gaveta, é necessário adicionar as
                {
//coordenadas X e Y de qr_area, de modo a poder conseguir encontrar os
códigos QR numa imagem do conteudo inteiro da caixa
                    qr_blocks[j].X += qr_area[i].X;
                    qr_blocks[j].Y += qr_area[i].Y;
                }

                //for (j = 0; j < blocks.Length; j++)
                //{
                //    ImageView-
er.Show_Image(imageCopy2.Copy(blocks[j]));
                //    ImageView-
er.Show_Image(imageEmgu.Copy(qr_blocks[j]));
                //}
```

```
                ////////////////////////////////////////////
                ////    DESCODIFICAÇÃO DOS CÓDIGOS QR     //
                ////////////////////////////////////////////

                barcode_text = new string[qr_blocks.Length];
//Declaração de um array de strings que vai receber o conteúdo do QR
code

                Decode_QR(qr_blocks, barcode_text, imageEmgu, im-
ageCopy);


                ////////////////////////////////////////////
                //      PREENCHIMENTO DOS BLOCOS          //
                ////////////////////////////////////////////

                Fill_Blocks(blocks, blocks_letters, blocks_numbers,
blocks_letters_left, blocks_letters_right, blocks_num_left,
blocks_num_mid, blocks_num_right, year_end, imageCopy3);


                ////////////////////////////////////////////
                //     OCR E REGISTO DE VARIAVEIS         //
                ////////////////////////////////////////////

                blocks_info = new User_Info[blocks.Length];
//Inicialização dos objetos blocks_info, agora com o número correto de
blocos
                Array.Resize(ref drawer_info[i], blocks.Length);
//Resize das 6 posições X de drawer_info, agora com o número correto
de blocos

                Perform_OCR(blocks, blocks_letters_left,
blocks_letters_right, blocks_num_left, blocks_num_mid,
blocks_num_right, imageCopy2, blocks_info);


                ////////////////////////////////////////////////
                ///      CORREÇAO DE ERROS COM QR CODES       ///
                ////////////////////////////////////////////////

                QR_Correction(blocks, blocks_info, barcode_text);


                ////////////////////////////////////////////////
                ///     CORREÇAO DE ERROS COM PROOFREADER     ///
                ////////////////////////////////////////////////

                Proofreader_Correction(blocks, blocks_info, im-
ageCopy2);


                ////////////////////////////////////////////////////
                ///     COPIA DE BLOCKS_INFO PARA DRAWER_INFO     ///
                ////////////////////////////////////////////////////
```

```
                for (x = 0; x < blocks.Length; x++)          //Cópia da
informação de blocks_info, cujos conteúdos são passageiros, para a sua
posição respetiva de drawer_info,
                {                                                    //onde se-
rá mantida com mais permanência
                    drawer_info[i][x] = blocks_info[x];
                }
            }


            ////////////////////////////////////////////////
            ///         REGISTO DE DADOS NO EXCEL          ///
            ////////////////////////////////////////////////

            Excel_Save(drawer_info, drawer_name, drawer_num);

            System.Media.SystemSounds.Beep.Play();
        }
    }
}
```

8.2.3 ImageClass (apenas locateBlocks)

```
/// <summary>
        /// Roughly locates each and every block, saves their coordi-
nates in a an array of Rects and returns them
        /// </summary>
        /// <param name="img"></param>
        /// <param name="dpi"></param>
        /// <returns></returns>
        public static Rectangle[] locateBlocks(Image<Bgr, byte> img,
int dpi)
        {
            unsafe
            {

                ////////////////////////////////////////
                //      DECLARAÇAO DE VARIAVEIS         //
                ////////////////////////////////////////

                MIplImage m = img.MIplImage;
                byte* dataPtr = (byte*)m.imageData.ToPointer();

                int width = img.Width;
                int height = img.Height;
                int nChan = m.nChannels; // number of channels - 3
                int padding = m.widthStep - m.nChannels * m.width; //
alinhament bytes (padding)

                int i, j, k, n = 0;
                int count = 0;
                int start = 0;
                int[] variance = new int[height];           // Array
```

```
de variancias
                Rectangle[] blocks = new Rectangle[100];    // Array
de rects que vai receber as coordenadas dos blocos -> inicializado a
100 por nao se saber quantos blocos vao ser reconhecidos
                int[] block_start_line = new int[100];      // Array
de coordenadas de linhas de inicio de bloco
                int[] block_end_line = new int[100];        // Array
de coordenadas de linhas de fim de bloco
                int[] rect_value_sum;

                ////////////////////////////////////////
                //      ENCHIMENTO DAS VARIANCIAS       //
                ////////////////////////////////////////

                for (i = 0; i < height; i++)     //vai ver da esquerda
para a direita
                {
                    for (j = 0; j < width; j++)        //de cima para
baixo
                    {
                        if((dataPtr+nChan)[0] != dataPtr[0])
//se o valor do proximo pixel for diferente do pixel atual
                        {
                            variance[i]++;
//o valor de variancia aumenta 1
                        }

                        dataPtr += nChan;
//avanca para a proxima coluna
                    }

                    dataPtr += padding;                              //
avanca para a proxima linha
                }


                ////////////////////////////////////////
                //    PROCURA DOS LIMITES SUPERIORES    //
                ////////////////////////////////////////

                for (i = 0, j = 0; i < height; i++)                  //
vai ver de cima para baixo
                {
                    if (variance[i] >= 10)                           //
o valor da variância da linha i tem que ser 10 ou mais para ser candi-
data a inicio de bloco
                    {                                                //
o valor 10 foi determinado por aproximaçao, utilizando os resultados
obtidos das primeiras analises as caracteristicas da imagem
                        for (n = 0; n < (dpi/30) && (i + n) != height;
n++)
                        {
                            if (variance[i + n] >= 10)
// verifica, enquanto i+n nao ultrapassar a altura da imagem,
                            {
// se as próximas dpi/30 linhas têm uma variância 10+, para evitar
falsos positivos
                                count++;
```

```
// por cada variância = 10+, incrementa count
                        }
                    }

                    if (count == (dpi/30))
// se count for dpi/30, i.e., se as proximas dpi/30 linhas tiverem uma
variancia 10+
                    {
                        block_start_line[j] = i - 3*(dpi/300);
// estabelece a linha de inicio de bloco, com um offset em funçao de
dpi/30 para centrar melhor o bloco
                        count = 0;                                  //
faz reset ao count, para usar na busca do próximo bloco
                    }
                    else
                    {                                               //
se count não chegar a dpi/30
                        count = 0;                                  //
count é reinicializado pela mesma razão acima
                        i++;                                        //
i é incrementado e faz-se continue para continuar a ir a procura de um
novo bloco
                        continue;
                    }

                    if (block_start_line[j] > height)           //
se o valor da linha for maior que height, majora no próprio height de
modo a impedir um valor out of bounds
                    {
                        block_start_line[j] = height;
                    }


                    ////////////////////////////////////////
                    //    PROCURA DOS LIMITES INFERIORES    //
                    ////////////////////////////////////////

                    block_end_line[j] = i + (dpi/5)-3*(dpi/300);
// o tratamento já foi feito no limite superior, pelo que basta adici-
onar um offset funçao de dpi para obter o limite inferior

// Cada bloco, para dpi = 300, tem apx. 60 pixels de altura, pelo que
se usa esse valor como base para os calculos acima

                    if (block_end_line[j] > height)
                    {
                        block_end_line[j] = height;             //
se o valor da linha for maior que height, majora no próprio height de
modo a impedir um valor out of bounds
                    }

                    j++;                                            //
incrementa j para passar ao próximo bloco
                    i += (dpi/5) - 2*(dpi/300);                 //
adiciona o offset(dpi) a i para passar a primeira linha após terminar
o bloco

                    if (i >= height)                                //
```

```
se i chegar a um valor acima de height, está out of bounds, pelo que
termina com o break
                            break;

                        }
                    }


                    ////////////////////////////////////////
                    //     RESIZE DOS ARRAYS DOS BLOCKS     //
                    ////////////////////////////////////////

                    int aid1 = block_start_line.Distinct().Count() - 1;
//inicializar 2 int's aid1 e aid2 para fazer o resize dos arrays
block_start_line e block_end_line
                    int aid2 = block_end_line.Distinct().Count() - 1;

                    Array.Resize(ref block_start_line, aid1);
// Tendo em conta os valores de aid1 e aid2, faz-se o resize dos ar-
rays de blocos, de modo a ocupar menos espaço
                    Array.Resize(ref block_end_line, aid2);
                    Array.Resize(ref blocks, aid1);

                    int length = block_start_line.Length;

                    Image<Bgr, byte> imageBlocks = img.Copy();

                    for (k=0; k<length; k++)
// com o length correcto de blocos por retirar, vamos passar cada blo-
co para o array
                    {
                        blocks[k] = new Rectangle(0, block_start_line[k],
width, block_end_line[k] - block_start_line[k]);  // Cada elemento de
blocks vai receber um rect com as coordenadas averiguadas acima
                    }

                    for (k = 0; k < length; k++)
// Devido à natureza aproximativa da definiçao dos limites dos blocos,
ha risco de existirem restos do QR code a esquerda
                    {
                        blocks[k].X = CheckArti-
facts(imageBlocks.Copy(blocks[k])) - 1;  // Assim sendo, chama-se a
funçao CheckArtifacts para verificar a existencia ou nao desses mesmos
restos, e tratar deles
                        blocks[k].Width = width - blocks[k].X;
// A width do rect terá que ser, forçosamente, alterada para refletir
a eventual mudança na linha de inicio do bloco
                    }

                    return blocks;
                }
            }
```